Guide to VMS Text Processing

Order Number: AA–LA13B–TE

June 1990

This manual contains tutorial information about the Extensible VAX Editor (EVE), the EDT text editor, and DIGITAL Standard Runoff (DSR).

Revision/Update Information: This document supersedes the *Guide to VMS Text Processing*, Version 5.2.

Software Version: VMS Version 5.4

digital equipment corporation
maynard, massachusetts

June 1990

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

CDA	DEQNA	MicroVAX	VAX RMS
DDIF	Desktop–VMS	PrintServer 40	VAXserver
DEC	DIGITAL	Q-bus	VAXstation
DECdtm	GIGI	ReGIS	VMS
DECnet	HSC	ULTRIX	VT
DECUS	LiveLink	UNIBUS	XUI
DECwindows	LN03	VAX	
DECwriter	MASSBUS	VAXcluster	digital™

The following is a third-party trademark:

PostScript is a registered trademark of Adobe Systems Incorporated.

ZK4528

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by Digital. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use Digital-supported devices, such as the LN03 laser printer and PostScript printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

Contents

TABLES

Preface

This manual contains tutorial information to help you get started using the Extensible VAX Editor (EVE), the EDT text editor, and the DIGITAL Standard Runoff (DSR) text formatter.

Intended Audience

This manual is intended for new users of EVE, EDT, and DSR, and for those users who want to review the features of these tools.

Document Structure

This manual is divided into three chapters.

- Chapter 1 explains how to use the EVE.
- Chapter 2 explains how to use the EDT editor.
- Chapter 3 explains how to use the DSR text formatter.

Associated Documents

For detailed information about EVE, which is the interface to the VAX Text Processing Utility (VAXTPU), see the *VMS EVE Reference Manual* and the *VAX Text Processing Utility Manual*. For information about using EVE on DECwindows, see the chapter on EVE in the *VMS DECwindows Desktop Applications Guide*.

For further information about the EDT editor, see the *VAX EDT Reference Manual*. For reference information about DSR, see the *VAX DIGITAL Standard Runoff Reference Manual*.

Conventions

Convention	Meaning
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
Return	In examples, a key name is shown enclosed in a box to indicate that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)

Convention	Meaning
. . .	In examples, a horizontal ellipsis indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
.	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
.	
.	
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[]	In format descriptions, brackets indicate that whatever is enclosed within the brackets is optional; you can select none, one, or all of the choices. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
red ink	Red ink indicates information that you must enter from the keyboard or a screen object that you must choose or click on. For online versions of the book, user input is shown in **bold**.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in online versions of the book.
italic text	Italic text represents information that can vary in system messages (for example, Internal error *number*).
UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter OPEN/READ), or they indicate the name of a routine, the name of a file, the name of a file protection code, or the abbreviation for a system privilege.
	Hyphens in coding examples indicate that additional arguments to the request are provided on the line that follows.

Convention	Meaning
numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 Editing Files with EVE

The Extensible VAX Editor (EVE) is a general-purpose text editor based on the VAX Text Processing Utility (VAXTPU), a high-performance, programmable text processor. Using EVE, you can create and edit new files or edit existing files. You can add text to a file and modify or format that text. EVE is interactive, so you see the changes to a file as you make them. EVE runs on VT300-, VT200-, or VT100-series terminals but does not run on hardcopy terminals. For information about using EVE on DECwindows, see the chapter on EVE in the *VMS DECwindows Desktop Applications Guide*.

Unlike the EDT editor (described in Chapter 2), EVE lets you display more than one buffer on the screen at a time and edit more than one file during the same editing session. EVE is easy to customize or extend using EVE commands and VAXTPU procedures.

EVE provides extensive online help, which provides information without disturbing your work. For more information about online help, see Section 1.4. EVE also provides two optional keypads: an EDT keypad or a WPS keypad. The EDT and WPS keypads provide most (but not all) EDT and WPS key functions. For more information about the EDT and WPS keypads, see Section 1.4.

1.1 Getting Started

To begin an editing session, invoke EVE with the DCL command EDIT/TPU. You can specify a file name when you enter the command or not. The session ends when you enter the EXIT or QUIT command. Exiting from EVE typically produces a new file or a new version of an existing file.

1.1.1 Beginning an Editing Session

You can start an editing session by creating a new file and inserting text into it during the session. You also can specify an existing file when you invoke EVE.

To begin an EVE editing session, enter the DCL command EDIT/TPU. If you want to edit or create a particular file, you can specify the file name on the command line. For example, to invoke EVE and create a new file named NEWFILE.DAT, enter the following command:

```
$ EDIT/TPU newfile.dat
```

This command produces a screen that appears as follows:

```
[End of file]
```

```
Buffer: NEWFILE.DAT                          | Write | Insert | Forward
Editing new file: could not find WORKDISK:[USER]NEWFILE.DAT
```

If you specified a file name on the command line, EVE inserts the text of the file you are editing into a temporary holding area called a **buffer**. The contents of the buffer are shown in an area of your screen called a **window**. EVE buffers exist only during the editing session. When you end an editing session, you can save your edits or discard them.

The end-of-file marker marks the end of an EVE buffer. It is visible only on the screen and does not become part of your file. When you add text to the buffer, the end-of-file marker moves down. Depending on the length of your terminal screen, the marker may not be visible when you view the beginning of a buffer that contains many lines of text.

A highlighted **status line** appears at the bottom of the EVE window and provides information about the buffer you are viewing in the window. The status line shows the buffer name, editing status (write or read-only), current mode (insert or overstrike), and current direction (forward or reverse).

If you invoke EVE and specify a file name on the command line, an informational message appears in the **message window** beneath the highlighted status line. The message states either that the file is a new file or that a certain number of lines were read from an existing file. During the editing session, EVE displays other messages in the message window.

To invoke EVE to edit an existing file named SCHEDULE.DAT, enter the following command:

```
$ EDIT/TPU schedule.dat
```

The contents of the file SCHEDULE.DAT appear in the main window on your screen.

When you invoke EVE to edit an existing file, you can use the asterisk wildcard character (*) as a substitute for some or all of the characters in the file name and file type. Using wildcards in EVE follows the same rules as using wildcards in DCL. You can use the percent wildcard character (%) as a substitute for a single character at a time, and you can use the ellipsis wildcard character ([. . .]) as a substitute for a directory specification. If only one match is made, the file is displayed on your screen. If more than one match is made, EVE displays a list of matching files and prompts you to provide a more complete file specification. If no match is made, EVE creates a buffer named MAIN.

Rather than name a file at the beginning of an editing session, you can invoke EVE with the command EDIT/TPU and then enter text into the buffer. For example, invoke EVE with the following command:

```
$ EDIT/TPU
```

This command produces a screen that appears as follows:

```
[End of file]


Buffer: MAIN                                  | Write | Insert | Forward
```

You can save the text by writing it to a file, using the WRITE FILE command described in Section 1.7. Alternatively, when you finish creating your file, EVE prompts for a file name with the following:

```
Type filename for buffer MAIN (press RETURN to not write it):
```

Type the name of the file and press the Return key to write out the buffer to a file.

You might want to use a command symbol to invoke EVE. For example, if you put the following statement in your login command file and then execute the file, you only need to type EVE to invoke the editor:

```
$ EVE :== EDIT/TPU
```

1.1.2 Ending an Editing Session

To end an EVE editing session, you can use the EXIT command or the QUIT command. With the EXIT command, EVE typically produces a new version of the edited file. With the QUIT command, EVE discards your edits. Regardless of how the editing session is ended, any existing versions of the files remain unchanged.

To save your edited text, use the EXIT command. You can enter the EXIT command by pressing the F10 key (on VT200-series or VT300-series terminals) or by pressing Ctrl/Z.

If you have modified the current buffer, EVE creates a new version of the file with the same file name and file type as the original version, with the version number incremented by 1. For example, if you use the EXIT command after modifying a file named FUN.DAT;1, the output file is named FUN.DAT;2:

```
Command:  EXIT
4 lines written to file WORKDISK:[USER]FUN.DAT;2
```

To end a session without saving your edits, use the QUIT command. Enter the QUIT command by pressing the Do key (PF4 on VT100-series terminals), typing QUIT at the *Command:* prompt, and pressing Return. For example, if you have modified a file named FUN.DAT and enter the QUIT command, the following display appears on your terminal screen:

```
Command:  QUIT
Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press Return if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press Return, and exit from the file using the EXIT command.

If you have modified buffers other than the current one, EVE asks if you want to save the contents of the other buffers. If you type Y, EVE creates a new version of an existing file, incrementing the version number by 1. EVE prompts for a file name if no file currently exists.

If no buffers have been modified, then EXIT and QUIT are the same. For example, if you use EVE to inspect a file without making edits, you can quit by pressing Ctrl/Z.

1.2 Entering EVE Commands

Once you have invoked EVE, you can enter EVE commands to edit text, move the cursor, and perform other operations. There are two ways to enter EVE commands: by pressing predefined keys and by typing the commands themselves.

In addition to the predefined keys, EVE offers two optional keypads. You can set an EDT keypad or a WPS keypad. The EDT and WPS keypads provide most of the EDT and WPS key functions, although they do not fully implement those editors. For more information about the EDT and WPS keypads, see Section 1.4.

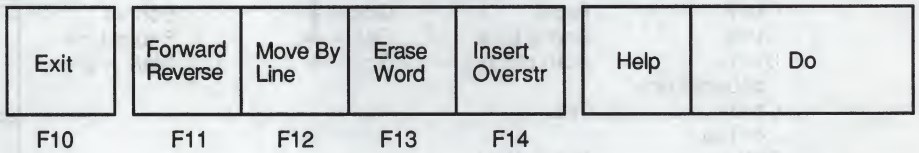
1.2.1 Using Defined Keys to Enter EVE Commands

EVE defines some keys by default. Each predefined key performs one editing command. The predefined keys on VT300-series and VT200-series terminals include the minikeypad (located between the main keyboard keys and the numeric keypad), certain function keys, and certain control key sequences. Figure 1–1 shows the predefined keys for the VT300-series and VT200-series terminal. On VT100-series terminals, EVE automatically defines most of the numeric keypad keys, the four arrow keys, and certain control keys. Figure 1–2 shows the predefined keys for the VT100-series terminal.

Control keys, arrow keys, and the Tab, Return, and <⌫ (Delete) keys have the same definition on all three types of terminal. For example, the Tab key on a VT100-series terminal does the same thing as the Tab key on a VT300- or VT200-series terminal.

Throughout this chapter, EVE editing keys are referred to by their names, rather than by their locations on the keyboard. For example, on a VT300-series or VT200-series terminal, two keys are defined as the Do key: the key located at the top of the editing keypad, labeled Do, and the PF4 key, located on the upper right of the numeric keypad. On a VT100-series terminal, the Do key and the PF4 key are the same. The PF4 key is also located at the upper right of the numeric keypad.

Figure 1–1 EVE Keys—VT200-Series and VT300-Series Terminals

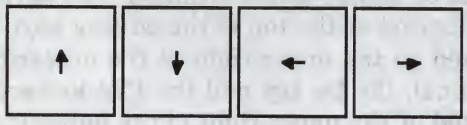


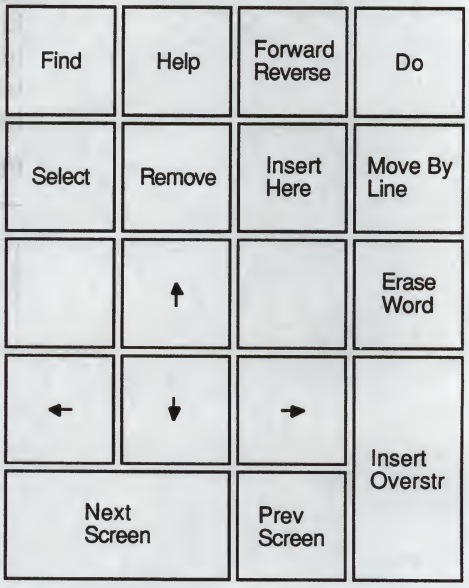
Key	Function
Ctrl/A	Change Mode
Ctrl/B	Recall
Ctrl/E	End of Line
Ctrl/H or Backspace	Start of Line
Ctrl/I or Tab	Tab
Ctrl/J or Line Feed	Erase Word
Ctrl/L	Insert Page Break
Ctrl/M or Return	Return
Ctrl/R	Remember
Ctrl/U	Erase Start of Line
Ctrl/V	Quote
Ctrl/W	Refresh
Ctrl/Z	Exit
PF4	Do
⌫	Delete

Gold Key Sequence	Function
Gold– ↑	Top
Gold– ↓	Bottom
Gold– →	End of Line
Gold– ←	Start of Line
Gold–Find	Wildcard Find
Gold–Insert Here	Restore
Gold–Remove	Store Text
Gold–Select	Reset
Gold–Prev Screen	Previous Window
Gold–Next Screen	Next Window
Gold–F13	Restore Word
Gold–Help	Help Keys

ZK–4036–GE

Figure 1–2 EVE Keys—VT100-Series Terminals





Key	Function	Gold Key Sequence	Function
Ctrl/A	Change Mode	Gold– ↑	Top
Ctrl/B	Recall	Gold– ↓	Bottom
Ctrl/E	End of Line	Gold– →	End of Line
Ctrl/H or Backspace	Start of Line	Gold– ←	Start of Line
Ctrl/I or Tab	Tab		
Ctrl/J or Line Feed	Erase Word		
Ctrl/L	Insert Page Break		
Ctrl/M or Return	Return		
Ctrl/R	Remember		
Ctrl/U	Erase Start of Line		
Ctrl/V	Quote		
Ctrl/W	Refresh		
Ctrl/Z	Exit		
PF4 or Do	Do		

ZK–4037–GE

You can define additional editing keys to enter commands or to perform editing operations that you frequently use. See Section 1.9 for information.

1.2.2 Typing EVE Commands

Besides using defined keys to enter commands, you can type commands at the *Command:* prompt. To type EVE commands, do the following:

1. Press the Do key. EVE displays the *Command:* prompt in the command window beneath the highlighted status line.
2. Type the EVE command after the prompt. The following example shows how to enter the EXIT command:

```
Command:  EXIT
```

3. Press Return or the Do key to enter the command.

You can correct typing mistakes on the EVE command line by pressing DCL line-editing keys. For example, use Ctrl/U to erase to the beginning of the line, Ctrl/E to move to the end of the line, and Ctrl/B to recall the last command entered. By default, the editing mode (insert or overstrike) of the EVE command line is the same as the editing mode of your terminal. (You can change the default with the DCL command SET TERMINAL prior to invoking EVE. Once in EVE, you can change the editing mode by pressing Ctrl/A.)

To save keystrokes when you are typing EVE commands, do the following:

- To recall the last EVE command you entered, press Ctrl/B. Pressing Ctrl/B again recalls the next to the last command and so forth. Continue pressing Ctrl/B until the command you want appears on your screen, and press Return to enter it.
- To abbreviate EVE command names, use the first letters of each command term. Make sure the abbreviation uniquely identifies the command. The following example shows what happens when the abbreviation you type is ambiguous. Enter the command RE.

```
Command:  RE
```

EVE displays a list of matching commands and prompts you for a choice. Type enough additional characters to ensure that the abbreviation is unique, and press Return to enter the command.

```
[End of file]

Buffer: MAIN                                | Write | Insert | Forward
Restore sentence    Restore line      Replace             Restore word
Restore selection   Reset             Restore character   Restore
Remove              Recall            Return              Refresh
Reverse             Repeat            Remember

Buffer: $CHOICES$
Command:  REF
Ambiguous command name:  RE
```

You can also abbreviate buffer names, file names, keynames, and mark names. Again, EVE provides a list of choices if you do not provide a unique abbreviation. (For more information about mark names, see Section 1.3.7.)

- To repeat an EVE command or keystroke a specified number of times, use the REPEAT command. Enter the REPEAT command and the number of times it is to be repeated. EVE repeats the next character or command you enter the specified number of times. For example, the following commands erase five words (the current word and the four previous):

```
Command:  REPEAT 5
Will repeat next command 5 times.
Command:  ERASE WORD
```

- To repeat the last command entered, press the Do key twice.

1.3 Editing Text

Once you know how to invoke EVE and how to enter commands, you can use EVE commands to create and edit files. Editing keys and commands let you move the cursor and perform editing operations such as moving, erasing, and restoring text.

1.3.1 Moving the Cursor

When editing files with EVE, you move the cursor where you want to perform an editing function. Therefore, the more quickly and efficiently you move the cursor through the text, the more time you save in your editing session.

The following tables show the EVE editing keys and commands that move the cursor. For more information about the Gold key combinations listed, see Section 1.9.3.

Key	What It Does
↑	Same as MOVE UP. Moves the cursor up one line. On VT100-series terminals, KP5 is also defined as MOVE UP.
↓	Same as MOVE DOWN. Moves the cursor down one line. On VT100-series terminals, KP2 is also defined as MOVE DOWN.
←	Same as MOVE LEFT. Moves the cursor one character or column to the left. On VT100-series terminals, KP1 is also defined as MOVE LEFT.
→	Same as MOVE RIGHT. Moves the cursor one character or column to the right. On VT100-series terminals, KP3 is also defined as MOVE RIGHT.
Ctrl/E or Gold–→	Same as END OF LINE. Moves the cursor to the end of the current line.
Ctrl/H or Gold–←	Same as START OF LINE. Moves the cursor to the beginning of the current line.
Gold–↑	Same as TOP. Moves the cursor to the top of the current buffer.
Gold–↓	Same as BOTTOM. Moves the cursor to the bottom of the current buffer.
Gold–Next Screen	Same as NEXT WINDOW. Moves the cursor to the last position the cursor occupied in the next window on your screen, if you are using two or more windows.
Gold–Prev Screen	Same as PREVIOUS WINDOW. Moves the cursor to the last position the cursor occupied in the previous window on your screen, if you are using two or more windows. For more information about Gold key combinations, see Section 1.9.3.

Command	What It Does
BOTTOM	Moves the cursor to the end of the current buffer. By default, EVE defines Gold–↓ as BOTTOM.
CHANGE DIRECTION	Changes the direction of the current buffer. The direction of the buffer is shown in the status line.
END OF LINE	Moves the cursor to the end of the current line. By default, EVE defines both Ctrl/E and Gold–→ as END OF LINE.

Command	What It Does
FORWARD	Default setting. Sets the direction of the current buffer to forward, that is, to the right and down. The direction of the buffer is shown in the status line.
LINE	Moves the cursor to the beginning of a line (specified by the line number).
MOVE BY LINE	In forward direction: moves the cursor to the end of the current line or, if the cursor is already at the end of a line, to the end of the next line. In reverse direction: moves the cursor to the beginning of the current line or, if the cursor is already at the beginning of a line, to the beginning of the previous line. On VT300-series and VT200-series terminals, EVE defines the F12 key as MOVE BY LINE, and on VT100-series terminals defines the Minus key on the keypad as MOVE BY LINE.
MOVE BY PAGE	Moves the cursor to the next or previous page break (form feed), depending on the current direction. If there is no page break in the current direction, the cursor moves to the bottom or top of the buffer.
MOVE BY WORD	In forward direction: moves the cursor to the beginning of the next word or, if the cursor is already at the end of a line, to the beginning of the next line. In reverse direction: moves the cursor to the beginning of the previous word or, if the cursor is already at the beginning of a line, to the end of the previous line.
NEXT SCREEN	Scrolls forward in the current buffer by the number of lines in the current window minus one. For example, if the current window is 12 lines long, the NEXT SCREEN command scrolls the cursor forward 11 lines. On VT300-series and VT200-series terminals, EVE defines the E6 key (Next Screen) as NEXT SCREEN, and on VT100-series terminals, defines the KP0 key on the keypad as NEXT SCREEN.
NEXT WINDOW or OTHER WINDOW	Moves the cursor to the next window on your screen, if there is one. The cursor appears in the last location it occupied in that window. EVE defines Gold–Next Screen as NEXT WINDOW.
PREVIOUS SCREEN	Scrolls backward in the current buffer by the number of lines in the current window minus one. For example, if the current window is 12 lines long, the PREVIOUS SCREEN command scrolls the cursor backward 11 lines. On VT300-series and VT200-series terminals, EVE defines the E5 key (Prev Screen) as PREVIOUS SCREEN, and on VT100-series terminals, defines the Period key on the keypad as PREVIOUS SCREEN.
PREVIOUS WINDOW	Moves the cursor to the previous window on your screen, if there is one. The cursor appears in the last location it occupied in that window. EVE defines Gold–Prev Screen as PREVIOUS WINDOW.
REVERSE	Sets the direction of the current buffer to reverse, that is, to the left and up. The direction of the buffer is shown in the status line.

Command	What It Does
SET CURSOR BOUND	Makes the cursor follow the flow of text. The cursor cannot move into an unused portion of the buffer. Similar to cursor behavior in EDT, WPS, and other editors.
SET CURSOR FREE	Default setting. Lets you move the cursor anywhere in the buffer and enter text there.
SET SCROLL MARGINS	Sets the top and bottom distances at which scrolling begins automatically as you move the cursor up and down. You specify these distances as numbers of lines or as a percentage of the window size. The default setting is 0, that is, scrolling starts when you move past the top or the bottom of the window.
SHIFT LEFT	Shifts the current EVE window to the left by the number of columns you specify. Using SHIFT RIGHT and SHIFT LEFT commands lets you view the undisplayed portion of long lines of text without having to change the width of the window or use 132-column mode. You can use the SHIFT LEFT command only if you have used the SHIFT RIGHT command.
SHIFT RIGHT	Shifts the current EVE window to the right by the number of columns you specify. Using SHIFT RIGHT and SHIFT LEFT commands lets you view the undisplayed portion of long lines of text without having to change the width of the window.
START OF LINE	Moves the cursor to the beginning of the current line. By default, EVE defines both Ctrl/H and Gold–← as START OF LINE.
TOP	Moves the cursor to the beginning of the current buffer (upper left corner). By default, EVE defines Gold–↑ as TOP.

The following example shows how to move the cursor through a buffer.

Invoke EVE and create the file SCHEDULE.DAT with the following command:

```
$ EDIT/TPU schedule.dat
```

The command produces the following screen:

```
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
Editing new file: could not find WORKDISK:[USER]SCHEDULE.DAT
```

EVE puts the cursor at the top of the buffer and waits for you to enter text. The [End of file] marker moves down in the buffer as you enter the following text into the file SCHEDULE.DAT:

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
```

Notice that the cursor is positioned at the end of the text you inserted. To move the cursor to the beginning of the file, use the TOP command.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
Command:  TOP
```

Press Ctrl/E to move the cursor to the end of the first line of text. Ctrl/E works the same way in EVE as it does in DCL.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
```

Enter the BOTTOM command to move the cursor to the end of the buffer.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
Command: BOTTOM
```

Press the up arrow key to move the cursor up one line to the fourth line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
```

Press the Change Direction key to change the current buffer direction to reverse. Press the Move By Line key to move the cursor to the beginning of the third line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Reverse
```

Enter the command LINE 1 to move the cursor to the beginning of the first line of the buffer.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
Command:  LINE 1
```

1.3.2 Entering Text

You can enter sections of text, entire files, and special nonprinting characters (such as control characters) into whichever buffer you are currently editing.

The following tables show the editing keys and EVE commands that you use while entering or inserting text:

Key	What It Does
Ctrl/A	Same as the CHANGE MODE command. Changes the current editing mode as shown in the highlighted status line. In insert mode, EVE inserts text at the character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position. On VT300-series and VT200-series terminals, EVE defines the F14 key as CHANGE MODE, and on VT100-series terminals, defines the Enter key on the keypad as CHANGE MODE.
Ctrl/V	Same as the QUOTE command. Lets you insert nonprinting characters or control codes. You can search for special characters using the Find key. First, press the Find key, then press Ctrl/V and the special character to be found, and activate the search by pressing Return.

Command	What It Does
CHANGE MODE	Same as Ctrl/A. Changes the current editing mode as shown in the highlighted status line. In insert mode, EVE inserts text at the current position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position. On VT300-series and VT200-series terminals, EVE defines the F14 key as CHANGE MODE, and on VT100-series terminals, defines the Enter key on the keypad as CHANGE MODE.
INCLUDE FILE	Inserts the contents of the specified file into the current buffer at the line above the cursor position. This is useful to combine files.
INSERT MODE	Sets the mode of the current bufffer to insert, as opposed to overstrike. In insert mode, EVE inserts text at the current position, moving existing text to accommodate the insertion.

Command	What It Does
OVERSTRIKE MODE	Sets the mode of the current buffer to overstrike, as opposed to insert. In overstrike mode, EVE overwrites text at the current position.
QUOTE	Same as Crtl/V. Enters a nonprinting character or a control code that you specify by a key press. You can quote a control code or other character when you enter a string for the FIND or REPLACE commands. For example, you can quote the Tab key to search for tab characters.

Before you begin typing text, look at the highlighted status line to check whether the buffer is in insert or overstrike mode. If the buffer is in insert mode, text is inserted at the cursor position, and text that already appears in the file moves to accommodate your insertions. If the buffer is in overstrike mode, text that you type at the keyboard is inserted at the cursor position, and the text that already appears in the file is overwritten as the cursor moves through it.

Press Ctrl/A to change from one mode to the other.

You can add text to your buffer in the following ways:

- Text—You can type characters at the keyboard. EVE adds the characters to the buffer at the current cursor position according to the current mode of the buffer (insert or overstrike).
- Files—You can add an entire file by pressing the Do key and entering the EVE command INCLUDE FILE. Type the file specification at the *File to include:* prompt and press Return. Regardless of the current mode (insert or overstrike) of the buffer, EVE inserts the entire contents of the specified file into the buffer just before the line where the cursor currently appears.

  You can use wildcards in the file specification. If there is more than one match for a file specification with a wildcard, EVE displays a list of choices and prompts you to provide a more complete file specification. If the specified file does not exist, EVE displays a message stating that it could not include the file.
- Special nonprinting characters—You can use the QUOTE command to add special nonprinting characters by pressing Ctrl/V followed by the special character. For example, to insert an escape character into the buffer, press Ctrl/V followed by Ctrl/[. The special character is added according to the current mode of the buffer (insert or overstrike).

The following example shows you how to insert text into a file, first in insert mode and then in overstrike mode. Invoke EVE to edit the existing file SCHEDULE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]
Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
```

Check the highlighted status line to ensure that EVE is in insert mode. Press Ctrl/A to change to insert mode if EVE is in overstrike mode. Move the cursor to the first letter *s* in the word *supervisor*, type *Engineering*, and press the space bar.

The word *Engineering* is inserted in your text buffer, and the rest of the text on the line moves to the right.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]
Buffer: SCHEDULE.DAT                    | Write | Insert | Forward
```

Now press Ctrl/A to change to overstrike mode. Move the cursor to the letter *S* in the word *Sally* and type *Peggy*.

The word *Peggy* is placed in the buffer, overwriting the word *Sally*.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Peggy
Work on Pascal program
[End of file]
Buffer: SCHEDULE.DAT                  | Write | Overstrike | Forward
```

1.3.3 Erasing and Restoring Text

With EVE, you can easily erase text or correct mistakes made during an editing session. If you erase text by mistake, you can restore the most recently erased text to its former location or, by moving the cursor, to another location. The following tables show the editing keys and EVE commands that erase and restore text:

Key	What It Does
<X or Delete	Erases the character to the left of the cursor. Same as the DELETE command. If pending delete is enabled, DELETE erases text in the select range and puts it into the Restore Selection buffer. For more information about using pending delete, see Section 1.3.4.
Ctrl/J	Same as ERASE WORD. Erases the current word or, if the cursor is between words, erases the next word. On VT300-series and VT200-series terminals, EVE defines the F13 key as ERASE WORD, and on VT100-series terminals, defines the Comma key on the keypad as ERASE WORD.
Ctrl/U	Same as ERASE START OF LINE. Erases characters left of the cursor to the start of the line.

Key	What It Does
Gold–Insert Here	Same as RESTORE. Reinserts, at the current position, the word or line that you just erased with an EVE command or editing key. For more information about Gold key combinations, see Section 1.9.3.
Gold–F13	Same as RESTORE WORD (except with the WPS keypad). Reinserts, at the current position, the word that you just erased. For more information about Gold key combinations, see Section 1.9.3.

Command	What It Does
DELETE	Erases the character to the left of the cursor. In insert mode, EVE inserts text at the current character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position. At the start of a line, DELETE erases the carriage return for the previous line (regardless of mode) and the current line moves up. If pending delete is enabled, DELETE erases text in the select range and puts it into the Restore Selection buffer. For more information about using pending delete, see Section 1.3.4.
ERASE CHARACTER	Erases the character the cursor is on. In insert mode, EVE inserts text at the current character position, moving existing text to accommodate the insertion. In overstrike mode, EVE overwrites text at the current position. If the cursor is at the end of the line, the carriage return is erased—regardless of the mode—and the next line moves up.
ERASE LINE	Erases from the current character to the end of the line, appending the next line to the end of the current line. If the cursor is at the end of the line, only the carriage return is erased and the next line moves up.
ERASE PREVIOUS WORD	Erases the previous word or the word the cursor is on. If the cursor is between words or on the first character of a word, the previous word is erased. If the cursor is in the middle of a word, all of that word is erased (same as ERASE WORD). If the cursor is at the start of a line, the carriage return at the end of the previous line is erased and the current line moves up.
ERASE START OF LINE	Erases the current line of text, starting with the character left of the cursor until the start of the line. If you are already at the start of a line, nothing is erased.

Command	What It Does
ERASE WORD	Erases the current word or, if the cursor is between words, erases the next word. Same as Ctrl/J. On VT300-series and VT200-series terminals, EVE defines the F13 key as ERASE WORD and, on VT100-series terminals, defines the Comma key on the keypad as ERASE WORD. If the cursor is at the end of the line, only the carriage return is erased and the next line moves up.
RESTORE	Reinserts, at the current position, the word, line, or sentence that you last erased with an EVE command or editing key. RESTORE does not restore single characters. EVE defines Gold–Insert Here as RESTORE.
RESTORE CHARACTER	Reinserts, at the current position, the character you last erased with an EVE command or editing key. In overstrike mode, the restored character replaces the character the cursor is on. In insert mode, the restored character is inserted at the cursor position and existing text moves to accommodate it.
RESTORE LINE	Reinserts, at the current position, the line that you last erased with an EVE command or editing key.
RESTORE SELECTION	Reinserts, at the current position, the text last erased with a pending delete operation.
RESTORE WORD	Reinserts, at the current position, the word that you last erased with an EVE command or editing key. EVE defines Gold–F13 as RESTORE WORD (except with the WPS keypad).

To erase text from your buffer, move the cursor to the text you want to erase, and press the appropriate editing key or type the appropriate EVE command.

The following example shows you how to erase and restore text. Invoke EVE to create the file RHYMES.DAT and enter the text shown in the following display:

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

Move the cursor to the letter *l* in the word *also*. Enter the ERASE LINE command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  ERASE LINE
```

EVE erases all characters from the letter *l* in *also* to the end of the line and appends the next line to the current line.

```
She rhymes with tree,
aand this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

Move the cursor to the letter *y* in the word *rhymes*. Enter the ERASE WORD command.

```
She rhymes with tree,
aand this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  ERASE WORD
```

EVE erases the word *rhymes* and shifts the remaining text to the left.

```
She with tree,
aand this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

Move the cursor to the second letter *a* on the second line. Enter the RESTORE LINE command.

```
She with tree,
aand this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  RESTORE LINE
```

EVE restores the last line that was erased, in this case, *lso with bee,*.

Move the cursor to the letter *w* in the word *with* on the first line. Enter the RESTORE WORD command.

```
She with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  RESTORE WORD
```

EVE restores the last word that was erased, in this case, *rhymes*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

Section 1.3.4 describes the functions of the SELECT and REMOVE commands, which can be used together to erase text from a buffer.

1.3.4 Moving Text from One Location to Another

Using EVE commands, you can select text for copying, moving, deleting, or other editing operations. For information on how to move text from one buffer to another, see Section 1.7. You can also select a rectangular area (a box) of text rather than a linear range of text to move, erase, or duplicate. For information about using box editing commands, see Section 1.3.5

The following tables describe the functions of the SELECT, REMOVE, and INSERT HERE commands as well as the STORE TEXT, COPY, CUT, PASTE, SET PENDING DELETE, RESTORE SELECTION, and SELECT ALL commands.

Key	What It Does
Insert Here	Same as the INSERT HERE or PASTE command. Inserts, at the current position, text that you removed or copied.
Remove	Same as the REMOVE or CUT command. Removes the text that is marked with SELECT or highlighted by FIND, and places it in the Insert Here buffer.
Select	Marks text (highlighting it in reverse video) from the initial cursor location to wherever you move the cursor. The text that is highlighted is called the **select range**. To cancel the selection, press the Select key again or use RESET.
Gold–Select	Same as RESET. Cancels any of the following and resets the direction of the buffer to forward: • Highlighting of a select or found range • A press of the Gold key (or Gold-n combination for a repeat count) • An incomplete or recalled command line, or Choices buffer display • The output of SHOW, SHOW DEFAULTS BUFFER, SHOW SUMMARY, or SHOW WILDCARDS, thereby returning you to the buffer you were working in For more information about Gold key combinations, see Section 1.9.3.
Gold–Remove	Same as the STORE TEXT or COPY command. Copies text that is marked with SELECT or FIND, putting it in the Insert Here buffer. Text that is copied is not removed from its original position. For more information about Gold key combinations, see Section 1.9.3.

Command	What it Does
INSERT HERE or PASTE	Inserts the text you copied or removed. By default, EVE defines the E2 key (Insert Here on the minikeypad on VT300-series and VT200-series terminals) and the KP9 key (on VT100-series terminals) as INSERT HERE.

Command	What it Does
REMOVE or CUT	Removes the text that was marked with SELECT or highlighted by FIND, and places it in the Insert Here buffer. By default, EVE defines the E3 key (Remove on the minikeypad on VT300-series and VT200-series terminals) and the KP8 key (on VT100-series terminals) as REMOVE.
RESET	Cancels any of the following and resets the direction of the buffer to forward: • Highlighting of a select or found range • A press of the Gold key (or Gold-n combination for a repeat count) • An incomplete or recalled command line, or Choices buffer display • The output of SHOW, SHOW DEFAULTS BUFFER, SHOW SUMMARY, or SHOW WILDCARDS, thereby returning you to the buffer you were working in
RESTORE SELECTION	Reinserts the text erased by a pending delete operation.
SELECT	Highlights text in reverse video from the initial cursor location to wherever you move the cursor. The text that is highlighted is called the **select range**. To cancel the selection, enter the SELECT command again or use RESET. By default, EVE defines the E4 key (Select on the minikeypad on VT200-series and VT300-series terminals) and the KP7 key (on VT100-series terminals) as SELECT.
SELECT ALL	Highlights all text in reverse video in the current buffer regardless of the cursor position. The text that is highlighted is called the select range. To cancel the selection, enter the SELECT command or use RESET. You can temporarily disable pending delete with the SELECT ALL command to avoid accidentally erasing all of the buffer.
SET NOPENDING DELETE	Default setting. Disables deletion of selected text when you use the <⌫ (Delete) key or type new text. If you select text in the buffer, typing new text adds characters to the select range and using the <⌫ (Delete) key erases only the character to the left of the cursor.
SET PENDING DELETE	Enables pending delete, which lets you quickly erase blocks of text. First, enable pending delete, then use the SELECT command to choose the text you want to erase. Erase the text by pressing the <⌫ (Delete) key (or any other typing key). To reinsert what you deleted, move the cursor to where you want the text and enter the RESTORE SELECTION command. The default is SET NOPENDING DELETE.
STORE TEXT or COPY	Copies text that was marked with SELECT or FIND, placing it in the Insert Here buffer. Text that is copied is not removed from its original position.

To select and then erase text when the buffer is set in a forward direction, do the following:

1 Place the cursor on the first character you want to erase.

2 Press the Select key.

3 Move the cursor to one character beyond the last character you want to erase. (In reverse direction, move the cursor to the last character, not one beyond.) The text to be erased is highlighted in reverse video. (If you decide not to remove text from the buffer, press the Select key again to cancel the selection.)

4 Press the Remove key. EVE deletes the highlighted text from your screen and places it in the Insert Here buffer.

You can insert the text in the Insert Here buffer at any cursor location by pressing the Insert Here key, or you can erase text permanently from your buffer by leaving it in the Insert Here buffer. You can insert the text contained in the Insert Here buffer any number of times at any cursor location until you select a new section of text and put that new text in the Insert Here buffer using the Remove key or the STORE TEXT (or COPY) command. The Insert Here buffer contains whatever text was last copied or removed.

You can also use any of the following EVE editing commands on a highlighted select range:

- BOX COPY
- BOX CUT, BOX CUT, INSERT, or BOX CUT OVERSTRIKE
- CAPITALIZE WORD
- CONVERT TABS
- DELETE (with pending delete enabled)
- FILL or FILL RANGE
- FIND NEXT
- FIND SELECTED
- LOWERCASE WORD
- OPEN SELECTED
- REMOVE or CUT
- SPELL (if DECspell is installed on your system)
- STORE TEXT or COPY
- UPPERCASE WORD
- Some EDT or WPS keys

The following example shows you how to erase and move text from one location to another using the Select, Remove, and Insert Here keys. Invoke EVE to edit the file RHYMES.DAT.

Move the cursor to the beginning of the second line of RHYMES.DAT and press the Select key.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Selection started. Press the Remove key when finished.
```

Press the down arrow key once. The second line of text is highlighted. Press the Remove key. The second line of text is removed from the current buffer.

```
She rhymes with tree,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Remove completed.
```

Press Return twice and then press the Insert Here key. The text in the Insert Here buffer is inserted at the current cursor location.

```
She rhymes with tree,


also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Remove completed.
```

The STORE TEXT command lets you copy text so you can insert it elsewhere. The COPY command is the same as the STORE TEXT command. You can substitute the COPY command wherever the STORE TEXT command is used in the following examples.

For the next example, use the following instructions to copy text when the buffer is set in a forward direction:

1. Move the cursor to the first line of text.
2. Press the Select key.
3. Press Ctrl/E to move the cursor to the end of the first line.
4. Enter the STORE TEXT command. The Insert Here buffer now contains a copy of the selected text.
5. Move the cursor to the line above *also with bee,*.
6. Press the Insert Here key.

```
She rhymes with tree,

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Store Text completed.
```

Move the cursor to the beginning of the first line of text. Use the Select key and then the Remove key to delete the first line of text.

1.3.5 Moving a Box of Text

You can edit text using rectangular areas—or **boxes**—as well as standard linear ranges. For example, you can select a box containing a list or columns in a table, and then cut and paste the box or perform some other editing operation on the box. Table 1–1 lists the commands for box editing.

Table 1–1 EVE Commands for Box Editing

Command	Usage
BOX COPY	Copies a box of text, without removing it, so you can paste it elsewhere.
BOX CUT	Cuts a box of text so you can paste it elsewhere, usually padding the area with spaces to keep the column alignment of text to the right of the box.
BOX CUT INSERT	Cuts a box, making text to the right of the box "collapse" to the left, closing the gap.
BOX CUT OVERSTRIKE	Cuts a box, padding the area with spaces to keep the column alignment of text to the right of the box.
BOX PASTE	Pastes a box of text you copied or cut, usually overwriting existing text.
BOX PASTE INSERT	Pastes a box, pushing existing text to the right.
BOX PASTE OVERSTRIKE	Pastes a box, overwriting existing text.
BOX SELECT	Selects a box of text. Typically, you start at the upper left corner of the box and move the cursor to where you want the lower right corner.
RESTORE BOX SELECTION	Puts back (undeletes) a box erased with pending delete, usually overwriting existing text.
SET BOX NOPAD	Disables padding and overstriking for box editing unless the buffer is in overstrike mode.
SET BOX NOSELECT	(Default.) Disables box selection, cutting, and pasting. Commands such as SELECT, COPY, and REMOVE use standard linear ranges. To edit boxes, use BOX commands.
SET BOX PAD	(Default.) Enables automatic padding and overstriking for box editing, regardless of the buffer mode.
SET BOX SELECT	Enables box selection, making commands such as SELECT, REMOVE, and INSERT HERE the same as the corresponding BOX commands, without having to redefine keys.

To select a box of text, do the following steps:

1 Put the cursor where you want to start the selection—typically, where you want the upper left corner of the box.

2 Enter the BOX SELECT command.

3 Move the cursor to where you want the diagonally opposite corner of the box—typically, moving from upper left to lower right.

As you move the cursor, text that you cross is highlighted in bold video (whereas a regular selection uses reverse video). The box is defined by diagonally opposite corners. If you move from upper left to lower right, the character that the cursor is on is *outside* the box, that is, the lower right corner of the box is left of the cursor.

You can then edit the box by using any of the editing commands that ordinarily work on a linear or a rectangular range. You need not redefine keys. See Section 1.3.4 for a list of EVE editing commands.

You can use FIND SELECTED or OPEN SELECTED if the selection does not cross lines. You can also use pending delete.

To cancel a box selection, repeat SELECT or BOX SELECT, or use RESET.

To cut and paste a box of text, do the following steps:

1 Select or find the text you want to cut. Typically, you use BOX SELECT.

2 Use BOX CUT to remove the text, or BOX COPY to copy the text without removing it.

  Note that with a box selection, you can use REMOVE or CUT to cut a box or STORE TEXT or COPY to copy a box. You need not redefine keys.

3 To paste the box you cut or copied, put the cursor where you want the upper left corner of the box, and then use BOX PASTE.

Cutting a box usually pads the area with spaces to keep the column alignment of text to the right of the box. Pasting a box usually overwrites existing text. Tab characters in the box, or that overlap the box, are converted to spaces to keep the column alignment of text.

The following example shows you how to select and then cut and paste a box of text.

Invoke EVE to create the file CITIES.DAT and enter the text shown in the following display:

```
      Rome     Paris    New York
      London   Tunis    Boston
      Tokyo    Bonn     Lisbon
[End of file]

Buffer: CITIES.DAT                    | Write | Insert | Forward
```

Move the cursor to the left of the letter *P* in the word *Paris*. Enter the BOX SELECT command.

```
      Rome    Paris    New York
      London  Tunis    Boston
      Tokyo   Bonn     Lisbon
[End of file]

Buffer: CITIES.DAT                          | Write | Insert | Forward
Command:  BOX SELECT
```

Move the cursor to the right of the second letter *n* in the word *Bonn*– the diagonally opposite corner of the box. The text that you cross is highlighted in bold video. Enter the BOX CUT command.

```
      Rome    Paris    New York
      London  Tunis    Boston
      Tokyo   Bonn     Lisbon
[End of file]

Buffer: CITIES.DAT                          | Write | Insert | Forward
Command:  BOX CUT
```

EVE removes the box of text. Move the cursor to the right of the column that begins with the words *New York*.

```
      Rome              New York
      London            Boston
      Tokyo             Lisbon
[End of file]

Buffer: CITIES.DAT                          | Write | Insert | Forward
```

Press the Do key and enter the BOX PASTE command.

```
COMMAND:  BOX PASTE
```

EVE pastes the box of text into a new column.

```
   Rome          New York      Paris
   London        Boston        Tunis
   Tokyo         Lisbon        Bonn
[End of file]

Buffer: CITIES.DAT                          | Write | Insert | Forward
```

If you are going to make several box edits—for example, in editing multicolumn tables and lists—use the SET BOX SELECT command, which effectively redefines several commands and keys as the corresponding BOX commands and makes other editing operations work on boxes instead of linear ranges.

SET BOX SELECT makes the following commands the same as BOX commands:

Command	Effect with SET BOX SELECT
INSERT HERE or PASTE	BOX PASTE
REMOVE or CUT	BOX CUT
RESTORE SELECTION	RESTORE BOX SELECTION
SELECT	BOX SELECT
STORE TEXT or COPY	BOX COPY

Thus, you can then select, cut, and paste a box by using the Select, Remove, and Insert Here keys, without having to redefine the keys.

You can use pending delete to erase a selected box of text. **Pending delete** refers to erasing a selection by simply typing new text (such as the space bar) or by using DELETE (typically, pressing the <⌫ key).

With a box selection, pending delete works like BOX CUT, usually padding the area with spaces to keep the column alignment of text to the right of the box. To erase a selection using pending delete, do the following steps:

1 To enable pending delete, use the SET PENDING DELETE command. The default setting is SET NOPENDING DELETE.

2 Select the text you want to erase. You can use SELECT or BOX SELECT. (You cannot use SELECT ALL.)

3 Type new text or use DELETE.

Thus, pending delete gives you an alternative way of cutting and pasting text, because pending delete does not use the Insert Here buffer or the DECwindows clipboard. For more information about pending delete, see the EVE online help topic called Pending Delete.

To put back (restore) the text you erased, do the following steps:

1 Put the cursor where you want to restore the text—that is, where you want the upper left corner of the box.

2 Use RESTORE BOX SELECTION or, if you used SET BOX SELECT, you can use RESTORE SELECTION (without having to redefine a key).

Restoring a box works like BOX PASTE, usually overwriting existing text. Using the SET BOX NOPAD command makes the effects of box editing depend on the mode that the buffer is in (insert or overstrike, as shown in the status line):

- With *insert mode*, cutting a box makes text to the right of the box "collapse" to the left, closing the gap. Tab characters to the right of the box are also converted to spaces to keep the column alignment as the text collapses to the left. This method is useful to remove columns from a table or list, such as in turning a four-column table into a two-column table. Pasting a box pushes existing text to the right, which is useful to add columns in the middle of a table.
- With *overstrike mode*, cutting a box pads the area with spaces to keep the column alignment of text to the right of the box. Pasting a box overwrites existing text. The effects are the same as with SET BOX PAD, which is the default setting.

The buffer mode also affects erasing a box with pending delete and restoring an erased box. It also affects the following EDT and WPS keys:

EDT keypad keys:

Append (KP9)
EDT Replace (Gold-KP9)

Subs (Gold-Enter)

WPS keypad keys:

WPS Copy (Gold-Minus)
WPS Cut (Minus or Remove)
WPS Paste (Comma, Gold-Comma, or Insert Here)
WPS Replace (Gold-' or Gold-")

1.3.6 Searching for Text

EVE commands let you search for and locate specific text in a buffer. You can search for and locate every occurrence of specific text, and you can search for text that is on a single line or is separated by a line break. Also, you can search for text using wildcards–either VMS or ULTRIX.

The following table describes the EVE commands that locate text in a buffer.

Command	What It Does
FIND	Searches the current buffer for the text string you specify and highlights the found text. The text that is highlighted is called the **found range**.
FIND NEXT	Searches for the string of text you last specified with the FIND, REPLACE, or WILDCARD FIND command.
FIND SELECTED	Searches for a string of text you have selected, rather than for a typed string.
SET FIND CASE EXACT	Enables case-exact searches. This is particularly useful to find or replace search strings in lowercase letters only.
SET FIND CASE NOEXACT	Default setting. Disables case-exact searches so that EVE finds any occurrence if you enter a search string in all lowercase letters.
SET FIND NOWHITESPACE	Default setting. Sets FIND and WILDCARD FIND commands to match tabs and spaces exactly as you specify in the search string, and to search for strings that are entirely on one line.
SET FIND WHITESPACE	Sets FIND and WILDCARD FIND commands to treat spaces, tabs, and up to one line break as "white space" so you can search for strings of two or more words regardless of how they are separated.
SET WILDCARD ULTRIX	Enables ULTRIX patterns for WILDCARD FIND.
SET WILDCARD VMS	Default setting. Enables VMS patterns for WILDCARD FIND.
SHOW WILDCARDS	Lists the wildcard patterns you can use with WILDCARD FIND, either VMS or ULTRIX, depending on your setting.
WILDCARD FIND	Searches for a pattern of text, using either VMS or ULTRIX wildcards, depending on your setting.

Use the FIND command to locate specific text in the current buffer. (By default, EVE defines the Find key on VT300-series and VT200-series

terminals and the PF1 key on VT100-series terminals as the FIND command.) Enter the FIND command. Then type the text you want to locate, which is called the **search string**. For example, enter the following commands to find the search string *rhymes with* in the forward direction.

```
Command:  FIND
Forward Find:  rhymes with
```

If the string is found, EVE moves the cursor to the beginning of the specified string.

If the search string contains all lowercase letters, EVE disregards the case of letters and locates any occurrence of the string. Thus, the search string *the* matches *the*, *THe*, and *thE*. If the search string contains one or more uppercase letters, EVE finds only the occurrences of the string in which the case of each letter is exactly the same. Therefore, the only match for the search string *tHis* is *tHis*.

If you want to match the case of your search exactly when searching for lowercase occurrences of a string, enter the SET FIND CASE EXACT command. Then when you enter a search string in all lowercase letters, EVE searches only for lowercase occurrences, skipping occurrences that contain uppercase letters. For example, the following commands enable case-exact searching and then find *digital* when it appears in lowercase only, skipping occurrences such as *Digital* or *DIGITAL*.

```
Command:  SET FIND CASE EXACT
Command:  FIND digital
```

The setting applies to the FIND, REPLACE, and WILDCARD FIND commands. You can save the setting in your section file or command file for future editing sessions. The default setting is SET FIND CASE NOEXACT.

EVE is sensitive to diacritical (accent) marks and locates only those occurrences of the string in which the diacritical marks are exactly the same. For example, in searching for *ë*, EVE does not find occurrences of *e*, *é*, *è*, or *ê*.

The current direction of the buffer determines whether EVE first searches in a forward or reverse direction.

If EVE cannot find the string in the current direction but finds it in the opposite direction, EVE prompts you to change direction. The following example shows the system response when the string *rhymes with* is found in the opposite direction from the search:

```
Forward Find:   rhymes with
Found in reverse direction.  Go there?
```

To search in the opposite direction, type *Y* and press Return. EVE moves the cursor to the first occurrence of the string in the opposite direction. The current direction in the highlighted status line does not change, however.

When EVE finds the search string, the editor highlights it and moves the cursor to the first letter of the string. You can use any one of the editing commands listed in Section 1.3.4 on a highlighted search string.

To cancel the highlighting, move the cursor off the search string or use the RESET command.

To find the next occurrence of the search string, press the Find key twice or enter the FIND NEXT command.

1.3.6.1 Using the FIND Command

The following example uses the existing file RHYMES.DAT to illustrate the use of the FIND command. When you invoke EVE to edit RHYMES.DAT, the cursor appears on the first letter of the first line of the buffer, and the current direction is forward.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

Press the Find key, type the letters *ree*, and press Return. The cursor moves to the letter *r* in the word *tree* and highlights the letters *ree*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Forward Find: ree
```

Press Find twice to find the next occurrence of the string *ree*. The cursor moves to the letter *r* in the word *three* and highlights the letters *ree*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Finding previous target: ree
```

When a search string is found and highlighted, you can use any command that works on a selected or found range except SPELL. Also, you cannot use a pending delete operation on a found range. For example, enter the UPPERCASE WORD command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  UPPERCASE WORD
```

The UPPERCASE WORD command changes the case of the highlighted letters from lowercase to uppercase.

```
She rhymes with tree,
also with bee,
and this one makes thREE.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

1.3.6.2 Finding a Selected String

Use the FIND SELECTED command to search for a string that is particularly complicated or is easily misspelled or mistyped. First, copy the text so that it is appears twice in the buffer. Then, move the cursor to the beginning of the string *rhymes with tree,* on the first line. Enter the SELECT command.

```
She rhymes with tree,
also with bee,
and this one makes three.

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  SELECT
```

Move the cursor to highlight the string.

```
She rhymes with tree,
also with bee,
and this one makes three.

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Move the text cursor to select text.
```

Press the Do key and type FIND SELECTED. The cursor moves to the next occurrence of the string *rhymes with tree,*. The selection is canceled and the found string is displayed in video bold.

```
She rhymes with tree,
also with bee,
and this one makes three.

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  FIND SELECTED
```

1.3.6.3 Using Wildcards in a Search

You also can use wildcards to search for text. EVE can search for text using either of two sets of patterns: VMS or ULTRIX. By default, EVE searches for text using the VMS wildcard patterns. The SHOW WILDCARDS command displays wildcard patterns for VMS, which include the asterisk (*) and percent sign (%).

First, remove the first three lines of text from the buffer, leaving just the original three lines of text displayed on the screen. Now, put the cursor at the beginning of the buffer. To search for text strings ending in *ee*, enter the command WILDCARD FIND **ee*.

```
She rhymes with tree,
also with bee,
and this one makes thREE.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  WILDCARD FIND *ee
```

EVE puts the cursor at the beginning of the line containing the *r* in *tree*.

If you let the WILDCARD FIND or FIND command prompt you for the search string, you can specify the direction of the search. Enter the WILDCARD FIND command and at the *Command:* prompt type the search string. You then can press a key defined as Change Direction, Forward, or Reverse. The key terminates the command (same as Return) and specifies the direction in which the search starts. (By default, EVE defines the F11 key on VT300-series and VT200-series terminals and the PF3 key on VT100-series terminals as the Change Direction key.)

1.3.6.4 Including White Space in a Search

Use the SET FIND WHITESPACE and SET FIND NOWHITESPACE commands to specify how the WILDCARD FIND and FIND commands treat the blank spaces between words, such as spaces, tabs, and line breaks. The SET FIND NOWHITESPACE command enables the commands to search for multiword strings on a single line, matching spaces and tabs exactly as they are found. The SET FIND WHITESPACE command enables the WILDCARD FIND and FIND commands to search for a string of two or more words regardless of how they are separated. It enables the FIND commands to search for a string that contains up to one line break and more than one space or tab between words. SET FIND NOWHITESPACE is the default search behavior.

The following examples show how the SET FIND WHITESPACE and SET FIND NOWHITESPACE commands work.

First, position the cursor on the *w* of the word *with* in the second line of the text, and press the Tab key. Then enter the SET FIND WHITESPACE command.

```
She rhymes with tree,
also       with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  SET FIND WHITESPACE
```

Then enter the FIND command specifying the search string *tree, also with*. EVE highlights the string that has *tree,* at the end of the first line and *also* at the beginning of the second line followed by a tab stop and then the word *with*.

```
She rhymes with tree,
also        with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Forward Find:  tree, also with
```

Now, enter the SET FIND NOWHITESPACE command.

```
She rhymes with tree,
also        with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  SET FIND NOWHITESPACE
```

Use the FIND command with the same search string, *tree, also with*. EVE responds that the search string cannot be found.

```
She rhymes with tree,
also        with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Forward Find: tree, also with
Could not find:  tree, also with
```

To find a search string when SET FIND NOWHITESPACE is enabled, it must be found in the buffer on a single line exactly as you enter it in the search string. In the following example, EVE highlights the text of the search string because the search string was found on a single line exactly as it was typed: *also* followed by a tab stop and *with bee,*.

```
She rhymes with tree,
also        with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Forward Find:  also        with bee,
```

1.3.7 Marking Locations in Text

The MARK and GO TO commands are useful when you edit a large file and know that you want to return to a specific location later in the editing session. The following table describes the MARK and GO TO commands:

Command	What It Does
MARK	Puts an invisible mark at the current cursor position. The mark exists for the rest of an editing session or until you change it but is not saved when you exit.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is found in another buffer, EVE moves the cursor to the other buffer and puts that buffer into the current window.

To mark your position, enter the MARK command followed by a label name of your choice. The label name can be one or more printable characters, including alphanumeric and punctuation characters, spaces, and tab characters. To return the cursor to the marked location, enter the GO TO command followed by the label name.

The following example shows you how to use the MARK command to mark a cursor position with the label name FIRST, and how to return to that cursor position using the GO TO command.

Move the cursor to the letter *b* in the word *bee*. Enter the MARK command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command: MARK
```

At the *Mark name:* prompt, enter *FIRST*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Mark name:  FIRST
```

Move the cursor to the letter *t* of the word *three*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Current position marked as FIRST
```

To return the cursor to the position labeled FIRST, enter the command GO TO followed by the mark name *FIRST*.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  GO TO FIRST
```

1.3.8 Replacing Text

The REPLACE command lets you replace a text string in the current buffer with another text string. This is useful if you have spelled a word incorrectly throughout a long file and you want to fix every occurrence of the misspelled word.

The following example shows you how to use the REPLACE command to replace every occurrence of the string *ee* with the string *oo*. Move the cursor to the top of the buffer. Press the Do key, type REPLACE, and press Return. Type *ee* at the highlighted *Old string:* prompt and press Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Old string:  ee
```

Type *oo* at the highlighted *New string:* prompt and press Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
New string:  oo
```

The cursor moves to the highlighted string *ee* in the word *tree*. Type *all* and press Return. All occurrences of the string *ee* are replaced with the string *oo*.

```
She rhymes with troo,
also with boo,
and this one makes throo.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Replace? Type Yes, No, All, or Quit:   all
```

EVE searches for the old string in the current direction and then, if necessary, in the opposite direction. If the old string is found in the opposite direction, EVE asks if you want to change direction and go there. If you want to go there, press Return. If you do not want to go there, type NO and press Return.

If the old string is found, EVE highlights the text and asks for one of the following choices. You need only type the first letter of the response and press Return.

Response	Effect
Yes	Replace this occurrence and find the next one. (Default. You can simply press Return.)
No	Skip this occurrence and find the next one.
All	Replace all the occurrences (no further prompting unless EVE finds an occurrence in the opposite direction).
Last	Replace this occurrence and stop here.
Quit	Skip this occurrence and stop here.

If the search covers the buffer more than once, EVE asks if you want to continue (so you can avoid replacing a string again when the old and new strings are similar). When the operation is finished, EVE tells you how many replacements were made.

The REPLACE command is case sensitive. If the old string has any uppercase letters, EVE searches for exact case matches. If the old string is all lowercase, EVE searches for any occurrence of the string regardless of its case. If the new string has any uppercase letters, EVE replaces the string exactly. If the old and new strings are all lowercase, EVE replaces the string according to the following rules:

- A capitalized version of the old string (first letter uppercase, others lowercase) is replaced by a capitalized version of the new string.
- An all-uppercase version of the old string is replaced by an all-uppercase version of the new string.
- Otherwise, the old string is replaced by an all-lowercase version of the new string.

The following table shows how EVE uses the case of the strings:

Old String	New String	Highlights	Replacements
butter	margarine	butter	margarine
		Butter	Margarine
		BUTTER	MARGARINE
		BUtteR	margarine
Butter	margarine	Butter	margarine
butter	Margarine	butter	Margarine
		Butter	Margarine
		BUTTER	Margarine

Old String	New String	Highlights	Replacements
		BUtteR	Margarine
Butter	Margarine	Butter	Margarine

If you want to find or replace only lowercase occurrences of a string, enter the SET FIND CASE EXACT command. Then if you enter a search string in all lowercase, EVE searches for only lowercase occurrences, skipping occurrences that contain uppercase letters. The setting applies to FIND, REPLACE, and WILDCASE FIND commands. The following table shows how EVE searches for and replaces only lowercase strings when you enter the SET FIND CASE EXACT command:

Old String	New String	Highlights	Replacements
butter	margarine	butter	margarine

The default case setting is SET FIND CASE NOEXACT.

1.4 Using Online Help

EVE has online help that supplies information on editing commands and keys without disturbing your work. You can get help by entering the HELP command or pressing the Help key.

To view a list of EVE commands and topics, enter the command HELP. Use the Previous Screen and Next Screen keys (the Period and KP0 keys on the keypad of a VT100-series terminal) to scroll through the list. To get information about a particular command, type a command name after the help prompt and press Return. The help text appears on the screen.

If you know the name of a specific command for which you want help, press the Do key, type HELP followed by the name of the command, and press Return. The help text for that command appears on the screen. For example, to get help on the MOVE BY LINE command, enter the command HELP MOVE BY LINE. The following help text appears on your screen:

```
MOVE BY LINE
Moves the cursor a line at a time in the current direction.
Keys:  EVE Default                          VT100 Keypad
       ------------------------------------------------
       F12                                  MINUS on keypad

Steps:

   1.  If necessary, set the direction to move in --- forward or reverse.

   2.  Use MOVE BY LINE (see key list above).

Usage notes:

o  In forward direction, moves to the end of the current line, or to the
   end of the next line, if any.
```

```
o  In reverse direction, moves to the start of the current line, or to
   the start of the next line, if any.

Related topics:

   CHANGE DIRECTION      END OF LINE      LINE      START OF LINE
```

You can abbreviate commands so long as the abbreviations are unambiguous. Often you can use the first letter of each command term. For example, you can use E C as the abbreviation for ERASE CHARACTER. For more information, see the online help topic Abbreviating.

Online help contains both general information and command information. For information about keypads, you can use the command HELP KEYPAD. More specific keypad help is also available. For example, if you elect to use the EDT or the WPS keypad, use the command HELP EDT DIFFERENCES or HELP WPS DIFFERENCES to get help on the difference between the EDT and WPS keypads within EVE and the original EDT and WPS keypads. Use HELP SET KEYPAD EDT or the command HELP SET KEYPAD WPS to get help on changing your default keypad.

To display a diagram of all defined keys, enter the command HELP KEYS or press the Help key. The diagram shows both the default editing keys and the keys you have defined for the minikeypad (LK201 keyboard), the main keypad, the keys F10 through F14 (LK201 keyboard), and the Gold key (described in Section 1.9.3).

After you display the keypad diagram, you can get help on a particular key by pressing the desired key. If you press a key to which you have assigned an EVE command, EVE provides the help text for that command.

1.5 Recovering from System Interruptions

EVE has recovery procedures for two types of system interruptions. You can recover a "lost" editing session with the journaling facility, using either buffer change journaling or keystroke journaling, and you can quickly remove extraneous characters that appear on your screen.

1.5.1 Using the Journal File

If you are editing a file and a system interruption occurs (that is, a break in communication between your terminal and the computer), journal files record your edits so that you can recover your "lost" editing session.

EVE provides two types of journaling and recovery:

- **Buffer change journaling** creates a separate journal file for each text buffer you create. This is the EVE default. Buffer change journaling works both on DECwindows and on character-cell terminals. You recover one buffer at a time, typically by using RECOVER BUFFER commands in EVE. You can recover buffers from different editing sessions. The recovery restores only your text—it does *not* restore settings, key definitions, or the contents of system buffers (such

as the Insert Here buffer or $RESTORE$ buffer) before the system failure.

- **Keystroke journaling**, as in previous versions of EVE, creates a single journal file for the editing session. Keystroke journaling works only on character-cell terminals—it does not work on DECwindows—and has other restrictions. The recovery recreates your editing session in a "player piano" fashion.

  For more information about keystroke journal recovery, refer to the *VMS EVE Reference Manual* or the *VMS DCL Dictionary*.

It is possible to have both types of journaling for an editing session, although there is usually no reason to do so. Generally, buffer change journaling is the better method to use, because it has fewer restrictions and because the recovery is usually quite fast.

Buffer change journaling creates a journal file for each text buffer. As you edit a buffer, the journal file records the changes you make, such as erasing, inserting, or reformatting text. When you exit from EVE or when you delete the buffer, the journal files are deleted. If a system failure interrupts your editing session, the journal files are saved. Your last few keystrokes before the system failure might be lost.

To recover buffers after a system failure, invoke EVE and use the RECOVER BUFFER commands for each buffer you want to recover.

Table 1–2 summarizes the EVE commands for buffer change journaling and recovery.

Table 1–2 EVE Commands for Buffer Change Journaling and Recovery

Command	Effect
RECOVER BUFFER	Recovers a specified buffer by using the journal file for the buffer. You can specify the name of the buffer or file you want to recover, or the name of the journal file for the buffer.
RECOVER BUFFER ALL	Recovers all your text buffers—one at a time—by using the journal files for the buffers, if there are any.
SET JOURNALING	Enables buffer change journaling for a buffer that you specify.
SET JOURNALING ALL	This is the default setting. Enables buffer change journaling for all your buffers.
SET NOJOURNALING	Disables buffer change journaling for a buffer you specify.
SET NOJOURNALING ALL	Disables buffer change journaling for all your buffers.

The name of the buffer change journal file derives from the name of the file or buffer being edited, and has the file type TPU$JOURNAL. For example, the text buffer JABBER.TXT would have the buffer change journal file name JABBER_TXT.TPU$JOURNAL. To find the name of the journal file for the current buffer, use the SHOW command.

There are two ways to recover your edits with buffer change journal files: using the /RECOVER qualifier on the command line when you invoke EVE, or using RECOVER BUFFER commands within EVE. For example, to recover the text of JABBER.TXT after a system failure, you could invoke EVE with the command:

```
$ EDIT/TPU jabber.txt/RECOVER
```

Alternatively, you can invoke EVE and use the following command to recover your text:

```
Command:  RECOVER BUFFER jabber.txt
```

If the buffer change journal file is available (in this case, a file named JABBER_TXT.TPU$JOURNAL), EVE shows the following information and asks if you want to recover that buffer:

Name of the buffer
Original input file for the buffer, if any
Output file for the buffer, if any
Source file for recovery, if any
Starting date and time of the editing session
Journal file creation date and time

If you want to recover the buffer, press Return. Otherwise, type NO and press Return.

If the buffer you want to recover exists—typically, the MAIN buffer—EVE first deletes that buffer and then does the recovery. If the buffer you want to recover has been modified, EVE prompts you whether to delete the buffer before recovering.

If you are unsure of the buffer names or journal file names, specify the asterisk wildcard, as follows:

```
Command:  RECOVER BUFFER *
```

EVE then displays a list of all your available journal files so you can choose the one you want. The list appears in an EVE system buffer named $CHOICES$ in a second window. For more information about this buffer, see the EVE online help topic called Choices Buffer.

For more information about buffer change journaling, see the EVE online help topic called Journal Files.

1.5.2 Refreshing the Screen

If extraneous characters, such as an operator message, appear on your terminal screen while you are editing, press Ctrl/W or use the REFRESH command to refresh the screen. The screen becomes blank, and then all characters are redrawn, less any extraneous characters.

1.6 Formatting Text

EVE provides commands that let you format your text by setting margins, tabs, and word wrap. You can center lines, take extra white space out of text, and insert page breaks. The following tables show the editing keys and text formatting commands and describe their functions:

Key	What It Does
Return or Ctrl/M	Inserts a carriage return at the current position either to start a new line of text or to terminate a command you are typing. On VT300- and VT200-series terminals, EVE also defines the Enter key as Return.
Tab or Ctrl/I	Inserts a tab character at the current position according to the tab modes and at the tab stops currently set.
Ctrl/L	Inserts a form-feed character at the current position to mark the beginning of a new page. A page break appears as a small double F (F_F) and is always on a line by itself. Same as INSERT PAGE BREAK.

Command	What It Does
CAPITALIZE WORD	Changes the case of a word, making the first letter uppercase, and the rest of the letters lowercase. Works on a range, box, or single word.
CENTER LINE	Centers the current line between the left and right margins. The cursor moves with the line, remaining on the same character as the line moves.
CONVERT TABS	Converts tab characters to the appropriate number of spaces in a box, a range, or the entire buffer.
FILL	Reformats the current paragraph, range, or box according to the margins of the buffer, so the maximum number of words fits on a line. When you fill a select range or found range, the FILL or FILL RANGE command does not reformat a line that begins with a page break, RUNOFF command, or DOCUMENT tag, but does reformat the other lines in the range. Filling a range does not delete blank lines. For more information about select range, see Section 1.3.4. For more information about found range, see Section 1.3.6.
FILL PARAGRAPH	Reformats the paragraph the cursor is in, according to the margins set for the buffer. When you fill a paragraph, the FILL command does not reformat a line that begins with a page break, RUNOFF command, or DOCUMENT tag, but does reformat the other lines in the paragraph.

Command	What It Does
FILL RANGE	Reformats the range or box according to the current margin settings. When you fill a select range or found range, the FILL or FILL RANGE command does not reformat a line that begins with a page break, RUNOFF command, or DOCUMENT tag, but does reformat the other lines in the range. Filling a range does not delete blank lines.
INSERT PAGE BREAK	Inserts a form-feed character at the current position to mark the beginning of a new page. A page break appears as a small double F (F_F) and is always on a line by itself. By default, Ctrl/L is defined as INSERT PAGE BREAK.
LOWERCASE WORD	Changes the current word, range, or box to lowercase.
PAGINATE	Inserts a "soft" page break for a 54-line page. A soft page break appears as a form feed followed by the null character—(F_F N_L). When the PAGINATE command is entered, EVE moves back to the previous page break (if any) then checks ahead for page breaks within the next 54 lines. If any soft breaks are found within those 54 lines, EVE removes them. EVE then moves down 54 lines, inserts a soft break, and puts the cursor on the next line. The soft break is inserted on a line by itself. If a hard page break (form feed only) is found within the 54 lines, EVE stops on the line after the hard break, in case you want to erase the break.
SET LEFT MARGIN	Sets the left margin in the current buffer. The left margin must be greater than 0 but less than the right margin. By default, the left margin is 1 (leftmost column).
SET RIGHT MARGIN	Sets the right margin for the current buffer. The right margin must be greater than the left margin. By default, the right margin is one less than the width. The width is typically 80, so the default margin is typically 79.
SET PARAGRAPH INDENT	Specifies the number of spaces to be added to or subtracted from the first line of paragraphs you create or reformat. The default is 0 (no indent).
SET TABS AT	Sets tab stops at the columns that you specify. The column numbers must be in ascending order and separated by spaces. By default, tab stops are set every eight columns. The command does not affect the hardware tab settings of your terminal.
SET TABS EVERY	Sets tab stops at the specified interval. By default, tab stops are set every eight columns. The command does not affect the hardware tab settings of your terminal.
SET TABS INSERT	Default setting. Changes the tab mode so that EVE inserts a tab character at the current column when you press the Tab key. The cursor and text move to the next tab stop.

Command	What It Does
SET TABS SPACES	Changes the tab mode to insert an appropriate number of spaces, rather than a tab character, when the Tab key is pressed. Previously existing tab characters are not affected.
SET TABS MOVEMENT	Changes the tab mode so the Tab key becomes a cursor-movement key. Pressing the Tab key moves the cursor to the next tab stop but does not insert a tab character.
SET TABS VISIBLE	Makes tab characters visible on the screen, appearing as a small H$_T$ (horizontal tab).
SET TABS INVISIBLE	Default setting. Makes tab characters invisible on the screen, appearing as white space.
SET WRAP	Default setting. Enables word wrapping at the right margin of the buffer. EVE starts new lines without your pressing Return or using the FILL command.
SET NOWRAP	Disables word wrapping at the right margin of the buffer. To start new lines, press Return or use the FILL command.
UPPERCASE WORD	Changes the current word, range, or box to uppercase.

1.6.1 Setting Margins

The following example shows how to use EVE commands to set margins. Invoke EVE to edit the existing file RHYMES.DAT. Press the Do key, type SET LEFT MARGIN 20, and press Return to set a left margin of 20. The text that already appears in the buffer does not change.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  SET LEFT MARGIN 20
```

Move the cursor to the end of the buffer and type the following new text: *Also with thee, and me.* Then press Return. The new text that you enter is inserted at the left margin of 20.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Left margin set to: 20
```

Reset the left margin to 1. Press the Do key, type SET LEFT MARGIN 1, and press Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Left margin set to: 1
```

The text that already appears in the buffer does not change, and when you insert new text, it is inserted at a left margin of 1. Now, set the right margin to 30 by pressing the Do key, and entering SET RIGHT MARGIN 30.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  SET RIGHT MARGIN 30
```

Enter the following new text in your file and notice that it wraps automatically to the next line at a right margin of 30.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Right margin set to: 30
```

To reset the right margin to 79, press the Do key, type SET RIGHT MARGIN 79, and press Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Right margin set to: 79
```

1.6.2 Filling Text

The following example shows how to fill a selected range of text and how to fill a paragraph. Using the existing file RHYMES.DAT, set the left margin to 5 and the right margin to 55.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]
Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  SET LEFT MARGIN 5
Command:  SET RIGHT MARGIN 55
```

Now, fill a range of text by selecting the first three lines of text (with the Select key) and entering the FILL command. Press the Do key, and enter the command FILL.

```
She rhymes with tree,
also with bee,
and this one makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]
Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  FILL
```

EVE fills the highlighted text between the left margin of 5 and the right margin of 55.

```
    She rhymes with tree, also with bee, and this one
    makes three.
                    Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]
Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

For EVE, a paragraph is bound by the following:

- Blank lines
- Top or bottom of the buffer
- Page break (form feed at the start of a line)
- DSR command at the start of a line
- VAX DOCUMENT tag at the start of a line

To reformat text in a paragraph, put the cursor anywhere in the text of a paragraph, press the Do key, and enter the command FILL PARAGRAPH.

```
She rhymes with tree, also with bee, and this one
    makes three.
                        Also with thee, and me.
And free, and fee, and see,
and brie, and any number of
other words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  FILL PARAGRAPH
```

EVE fills the the paragraph according to the margins set for the buffer, in this case 5 and 55.

```
    She rhymes with tree, also with bee, and this one
    makes three. Also with thee, and me. And free, and
    fee, and see, and brie, and any number of other
    words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

1.6.3 Indenting Paragraphs

The following example shows how to add or subtract spaces from the first line of a paragraph. First, using the file RHYMES.DAT, set the left margin to 1, set the right margin to 40, and fill the paragraph. The buffer appears as follows:

```
She rhymes with tree, also with bee, and
this one makes three.  Also with thee,
and me.  And free, and fee, and see, and
brie, and any number of other words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  FILL PARAGRAPH
```

To indent the first line of each paragraph five spaces press the Do key, type SET PARAGRAPH INDENT 5, and press Return.

```
She rhymes with tree, also with bee, and
this one makes three.  Also with thee,
and me.  And free, and fee, and see, and
brie, and any number of other words.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  SET PARAGRAPH INDENT 5
```

When you create a new paragraph, EVE indents the first line of the paragraph five spaces.

```
She rhymes with tree, also with bee, and
this one makes three.  Also with thee,
and me.  And free, and fee, and see, and
brie, and any number of other words.

     This new paragraph is now indented
five spaces from the left margin.  All
subsequent paragraphs will also be
indented by five spaces.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

You can also use the SET PARAGRAPH INDENT command to create a "hanging" paragraph. A hanging paragraph can be used to create a bulleted list.

Using the file RHYMES.DAT, set the left margin to 5, set the right margin to 55, and fill the paragraph in the buffer as follows:

```
She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
    words.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  FILL PARAGRAPH
```

The indentation of the paragraph is set relative to the current left margin. To create a hanging paragraph, you can specify a negative value to the SET PARAGRAPH INDENT command so long as the algebraic sum of the paragraph indent and the left margin is at least 1.

Press the Do key, and enter SET PARAGRAPH INDENT –3.

```
She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
    words.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  SET PARAGRAPH INDENT -3
```

All subsequent paragraphs start at column 2.

```
She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
    words.

This new paragraph starts at column 2.  All other
    paragraphs you create will start at column 2 as
    well.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward

Paragraph indent set to: -3 (from left margin).
```

1.6.4 Centering Text

To center a line of text, put the cursor anywhere on the line you want to center. For example, to center the text *words* in the last line of RHYMES.DAT, put the cursor on the *w*. Press the Do key, and enter the command CENTER LINE.

```
She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
    words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  CENTER LINE
```

The command centers the line of text between the current left and right margins; in this example, the left margin is set at 5 and the right margin is set at 55. The cursor moves with the line, remaining on the character *w* as the line is centered.

```
    She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
                                  words.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

1.6.5 Changing Case

The EVE commands to change the case of text—CAPITALIZE, UPPERCASE, and LOWERCASE—work on a box, range, or single word.

To change the case of the first line of text, move the cursor to the word *She* and press the Select key. To mark the end of the selection, move the cursor to the end of the line after *one*, press the Do key, and enter the command UPPERCASE.

```
She rhymes with tree, also with bee, and this one
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
                        words.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
Command:  UPPERCASE
```

EVE changes the case of the select range to uppercase.

```
    SHE RHYMES WITH TREE, ALSO WITH BEE, AND THIS ONE
    makes three.  Also with thee, and me.  And free,
    and fee, and see, and brie, and any number of other
                        words.
[End of file]

Buffer: RHYMES.DAT                    | Write | Insert | Forward
```

To change the case of a particular word, put the cursor on the word, press the Do key, and enter the appropriate case-changing command. After EVE changes the case of the word, the cursor moves to the next word or to the end of the line. If the cursor is between two words, the case of the word to the right of the cursor is changed and the cursor moves to the next word.

1.7 Using Buffers

Buffers are storage areas that exist only during an editing session. The following table describes the EVE commands used to create, manipulate, and delete buffers. For more information about using EVE commands, see the *VMS EVE Reference Manual*.

Command	What It Does
BUFFER	Puts the specified buffer into the current window and moves the cursor to the last location it occupied in that buffer. If the specified buffer does not exist, creates a new buffer.
DELETE BUFFER	Deletes a buffer you specify by name.
GET FILE or OPEN	Puts the specified file into the current EVE window, creating a new buffer if necessary. If the file exists, EVE copies it into a new buffer in the current window. If the file does not exist, EVE creates a new, empty buffer, using the file name and file type for the buffer name. If there already is a buffer by that name, EVE asks for a different name to use.
GO TO	Returns the cursor to the location labeled by the MARK command. If the labeled location is found in another buffer, EVE moves the cursor to that buffer and puts it into the current window. (Section 1.7.3 explains how to use multiple buffers in an editing session.)
INCLUDE FILE	Inserts the contents of the specified file into the current buffer at the line above the cursor location. This is useful to combine files.

Command	What It Does
NEW	Creates a new buffer named MAIN and puts it into the current window. If the buffer MAIN already exists, EVE asks for a name for the new buffer.
NEXT BUFFER	Puts the next buffer (if one exists) into the current window and moves the cursor to the last position it occupied in that buffer. This command lets you move from one buffer to another without specifying a buffer name.
OPEN SELECTED	Opens a file whose name you have selected or found. This command is the same as using the GET FILE or OPEN command without having to type the file name.
REMOVE or CUT	If you are in the Buffer List buffer, same as DELETE BUFFER. Use the REMOVE command as follows to delete a buffer without typing the buffer name: enter the SHOW BUFFERS command (which puts you in the Buffer List buffer), move the cursor to the name of the buffer you want to delete, and enter the REMOVE command.
SAVE FILE	Writes the contents of the current buffer to the file associated with the buffer without ending the editing session. If you do not specify a file name with the SAVE FILE command, EVE prompts you for an output file specification. Similar to WRITE FILE.
SAVE FILE AS	Writes the contents of the current buffer to the file you specify without ending the editing session. Thus, if you are editing a file named FIRST.DAT you can save it as SECOND.TXT. This command does not change the name of the buffer. It does, however, associate the buffer with the file you name so any subsequent SAVE FILE or WRITE FILE commands or an EXIT command write the buffer to the file you named. This command requires you to supply a file specification.
SELECT or RETURN	If you are in the Buffer List buffer, selects the buffer you specify. Use the SELECT command as follows to select a buffer without typing the buffer name: enter the SHOW BUFFERS command, move the cursor to the name of the buffer you want to select, and enter the SELECT command.
SET BUFFER	Lets you specify the editing status of the buffer: whether the buffer can be modified or can be written to a file when you exit from EVE.
SHOW	Displays information about the buffers you have created during the editing session. If more than one buffer is active in your editing session, the SHOW command displays information about the buffer you are currently editing. For information about the other active buffers, press the Do key. To resume editing, press any other key.
SHOW BUFFERS	Lists the buffers you have created during an editing session. You can move the cursor through the list and specify a particular buffer for viewing by using the Select key.

Command	What It Does
SHOW DEFAULTS BUFFER	Shows information, such as margins, tab stops, direction, mode, and maximum lines, about the EVE system buffer named $DEFAULTS$. These are the default settings used when you create new buffers.
SHOW SYSTEM BUFFERS	Lists the system buffers created by EVE, such as the Message buffer, Help buffer, Insert Here buffer, and $RESTORE$ buffer. You can move the cursor through the list and specify a buffer for viewing by using the Select key.
WRITE FILE	Writes the contents of the current buffer to the file associated with the buffer or to the file you specify on the command line without ending the editing session. If the current buffer does not have a file specification associated with it, EVE prompts you for an output file specification. Similar to SAVE FILE.

When you edit an existing file, EVE reads the contents of the file into a buffer. The highlighted status line contains the name of the buffer, its editing status (read-only or write), editing mode (insert or overstrike), and direction (forward or reverse).

To display more information about the current buffer, enter the SHOW command. The information displayed includes whether the buffer has been modified, in addition to the following:

- Buffer name
- Names of the input, output, and buffer change journal files
- Current mode and direction
- Number of lines
- Margin- and screen-width settings
- Paragraph indent
- WPS word wrap
- Wrap indent
- Tab stop

If more than one buffer is active during an editing session, EVE prompts you to press the Do key to get information about other buffers.

To delete a buffer, enter the DELETE BUFFER command, and specify the name of the buffer you want to delete. If the buffer is empty or unmodified, EVE deletes it. If, however, the buffer has been modified, EVE prompts you for a choice. For example, the following command requests deletion of the modified buffer MYFILE.TXT:

```
Command:  DELETE BUFFER myfile.txt
That's a modified buffer.  Type delete_only, write_first, or quit:
```

The buffer name must be typed in full; no abbreviations are allowed.

The following table lists the choices you can enter:

Keyword	Effects
DELETE_ONLY	Deletes the specified buffer.
WRITE_FIRST	Writes out (saves) the specified buffer, then deletes it.
QUIT	Default choice. The buffer is not deleted.

If you are viewing a buffer that you want to delete, EVE replaces the buffer with the oldest buffer existing in the editing session.

Use the SET BUFFER command to change the editing status of the buffer, that is, whether the buffer can be modified and whether the buffer will be written to a file after you exit from EVE. You can specify one of the following keywords for each command:

Keyword	Effects
MODIFIABLE	Default setting. The buffer can be modified. Also restores the previous mode of the buffer (insert or overstrike).
READ_ONLY	The buffer is *not* saved (written out) on exiting, even if it has been modified (opposite of WRITE). Also sets the buffer to unmodifiable. However, you can set it to modifiable.
UNMODIFIABLE	The buffer cannot be modified. Also overrides the mode of the buffer (insert or overstrike).
WRITE	Default setting. The buffer is saved (written out) on exiting if it has been modified (opposite of READ_ONLY). If a buffer is read-only or unmodifiable, SET BUFFER WRITE makes it modifiable and restores its previous mode (insert or overstrike).

By default, buffer status is set to MODIFIABLE and WRITE. This allows you to change the contents of a buffer and save the changed buffer in a file.

To change the status of a buffer so that its contents cannot be inadvertently changed, set the buffer to READ_ONLY (which implies unmodifiable) with the following command:

```
Command:  SET BUFFER READ_ONLY
```

To change the status of a buffer so it becomes a temporary storage area (a "scratchpad"), set the buffer to READ_ONLY and MODIFIABLE with the following commands:

```
Command:  SET BUFFER READ_ONLY
Command:  SET BUFFER MODIFIABLE
```

You then can edit the buffer, but it will not be saved when you exit from EVE.

1.7.1 Listing Buffers

To display a list of all the buffers you have created during an editing session, enter the SHOW BUFFERS command. The following example shows the results of a SHOW BUFFERS command.

```
Buffer name                    Lines  Attributes

 FIRST_DRAFT.TXT                1234  Modified
 MAIN                             12  Modified
 MYBUFFER                         57  Modified
 SAMPLE.TXT                      592  Modified
 NOTMUCHHERE                       0
```

```
Buffer: BUFFER LIST      Use SELECT to view or REMOVE to delete buffers
```

You can scroll through the list and specify any buffers you want to delete or the buffer you want to view. To delete a buffer, move the cursor to the buffer name and press the Remove key.

```
Buffer name                    Lines  Attributes

 FIRST_DRAFT.TXT                1234  Modified
 MYBUFFER                         57  Modified
 SAMPLE.TXT                      592  Modified
 NOTMUCHHERE                       0
```

```
Buffer: BUFFER LIST      Use SELECT to view or REMOVE to delete buffers
Deleted buffer: MAIN
```

To display a buffer in your current window, move the cursor to the buffer name and press the Select key.

```
[End of file]
Buffer: NOTMUCHHERE                          | Write | Insert | Forward
```

These applications of the Select and Remove keys apply only when you are viewing a list of buffers.

To display a list of all buffers that EVE has created, enter the SHOW SYSTEM BUFFERS command. You can scroll through the list and specify the buffer you want to view by moving the cursor to the buffer name and pressing the Select key. EVE puts the buffer in your current window.

Note: Do not delete system buffers such as Insert Here, Messages, $RESTORE$, or $DEFAULT$, because these buffers are necessary for some commands to work properly.

1.7.2 Displaying the Messages Buffer

EVE uses the message window, which appears at the bottom of the screen, to communicate error and informational messages during an editing session. The message window displays the last message in the Messages buffer.

You can display these messages with the BUFFER command. To display the contents of the Messages buffer, enter the command BUFFER MESSAGES. To return to the buffer you were editing, enter the BUFFER command followed by the name of the appropriate buffer. The following example shows how to return to the buffer named RHYMES.DAT.

```
4977 lines read from file USER$:[SMITH.TEST]SECOND-DRAFT.TXT;62
Executing commands in initialization file: USER$:[SMITH]EVE$INIT.EVE;2
Right margin set to: 80
Tab mode set to insert spaces.

Left margin set to: 9

Move the text cursor to select text.

Remove completed.


Current position marked as: here

Move the text cursor to select text.

Store Text completed.

Going to mark: here

Buffer: MESSAGES                    | Read-only| Insert | Forward
Command: BUFFER RHYMES.DAT
```

Alternatively, you can enter the SHOW BUFFERS command to display the buffers you have created and use the Select key to choose a buffer.

1.7.3 Editing Multiple Buffers

During an editing session, you can use several buffers if you want to edit more than one file or if you want temporary storage areas for manipulating blocks of text.

You can create a new buffer using one of the following commands: GET FILE or OPEN, OPEN SELECTED, or BUFFER. To create a new buffer with a file that already exists, enter the GET FILE (or OPEN) command and the name of the file you want to copy to the new buffer. You can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the ellipsis wildcard ([. . .]) as a substitute for a directory specification.

Or you can create a new buffer using the OPEN SELECTED command as follows: put the cursor on the name of the file you want to open, then enter the OPEN SELECTED command.

To put a specific buffer into the current EVE window, enter the BUFFER command and the name of the buffer you want to put in the current window. You cannot use wildcard characters in buffer names (that is, * and % are treated as literal characters in a buffer name). If the buffer you specify does not already exist, EVE creates a new buffer.

If the specified file exists, EVE reads the contents of the file into a new buffer and displays the buffer in the current window. If there is more than one match for a file specification with a wildcard, EVE displays a list of choices in the $CHOICES$ buffer and prompts you to provide a more complete file specification. Note that EVE will open the first file it matches if you use a search list or an ellipsis wildcard ([. . .]). Otherwise, EVE creates an empty buffer and displays the buffer in the current window.

To change the buffer in the current window, press the Do key, type BUFFER and the name of the buffer you want to display on the screen, and press Return. If you forget a buffer name, enter the SHOW BUFFERS command to display the names of active buffers in your editing session and specify a buffer with the Select key.

The following example shows how to use two buffers to edit two files during an EVE editing session. This example assumes that the original versions of files RHYMES.DAT and SCHEDULE.DAT exist in your current default directory. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
3 lines read from WORKDISK:[USER]RHYMES.DAT
```

Press the Do key, type the command GET FILE SCHEDULE.DAT, and press Return to create a new buffer that contains the most recent version of SCHEDULE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                        | Write | Insert | Forward
```

Now that you have two buffers, practice copying from one buffer to another. First work with buffer SCHEDULE.DAT.

Place the cursor on the letter *R* in the word *Read* and press the Select key. Press the down arrow key once and the third line of SCHEDULE.DAT is highlighted.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                        | Write | Insert | Forward

Move the text cursor to select text.
```

Press the Remove key to place the selected line in the Insert Here buffer. The text from SCHEDULE.DAT remains in the Insert Here buffer until you overwrite it with other removed or copied text. Now switch to the other buffer to resume editing RHYMES.DAT. Press the Do key, type the command BUFFER RHYMES.DAT, and press Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

Position the cursor at the top of the buffer and press the Insert Here key. The text from the Insert Here buffer that was removed from the buffer SCHEDULE.DAT is inserted in the top of the buffer RHYMES.DAT.

```
Read and review memo from Sally
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

If you exit from an editing session in which you have modified multiple buffers, EVE writes the contents of the current buffer to a file and then asks you whether you want to write each of the other modified buffers to a file.

1.7.4 Reading and Writing Files

There are four ways to read a file into an EVE buffer:

- Invoke EVE with a file specification.
- Enter the INCLUDE FILE command and the name of the file you want to include. EVE reads the entire contents of the file into the buffer just before the line where the cursor is located. Using the INCLUDE FILE command does not change the name of the buffer on the status line.
- Enter the GET FILE or OPEN command and the name of the file you want to use. Either command creates a new buffer and reads the contents of an existing file into the buffer. The name of the buffer on the status line is the same as the file name you specify with the GET FILE or OPEN command. (See Section 1.7.3.)
- Select or find a file name, then enter the OPEN SELECTED command.

To write the contents of the current buffer to a file, enter the WRITE FILE command. You can include a file specification with the WRITE FILE command. If you do not include a file specification, EVE writes the file using the input file specification. If you created the current buffer with the BUFFER or NEW command, EVE prompts you for a file specification to which it writes the file.

The following example shows how to write out a buffer using the output file associated with the buffer.

```
Command:  GET FILE RHYMES.DAT
      .
      .
      .
Command:  WRITE FILE
3 lines written to WORKDISK:[USER]RHYMES.DAT;2
```

If you have used the WRITE FILE command in an editing session and you experience a system interruption, see Section 1.5 for information on recovering the editing session.

1.8 Using Windows

During an EVE editing session, the buffer you are editing is displayed on the screen in a window. A highlighted status line appears at the bottom of the window identifying the name, current editing mode, and current direction of the buffer.

EVE lets you view more than one window on your terminal screen at the same time. For example, you can have two windows on the terminal screen to view and edit different sections of the same buffer.

The following tables describe EVE keys and commands used to create and manipulate windows:

Key	What It Does in a Window Environment
Gold–Next Screen	Puts the cursor in the next (or other) window. Same as the NEXT WINDOW command. For more information about Gold key combinations, see Section 1.9.3.
Gold–Prev Screen	Puts the cursor in the previous (or other) window. Same as the PREVIOUS WINDOW command. For more information about Gold key combinations, see Section 1.9.3.

Command	What It Does in a Window Environment
DELETE WINDOW	Deletes the current window, if you are using more than one window.
ENLARGE WINDOW	Enlarges the current window by a specified number of lines. For example, ENLARGE WINDOW 5 enlarges the window by five lines. The adjacent window shrinks accordingly.
NEXT WINDOW or OTHER WINDOW	Puts the cursor in the next (or other) window.
ONE WINDOW	Restores the current window as a single, large window. EVE deletes all other windows from the screen. The buffers associated with those windows are not deleted.
PREVIOUS WINDOW	Puts the cursor in the previous (or other) window.

Command	What It Does in a Window Environment
SET WIDTH	Sets the width of lines displayed on the screen. Specify width as a positive integer. By default, the screen width is your terminal setting. (It is typically 80 columns.) If the width is set greater than 80, EVE sets the terminal to 132-column mode for the current editing session. When you exit from EVE, the terminal is restored to the default setting. Setting the width changes the display of text in all windows.
SHIFT LEFT	Moves the current window to the left a specified number of columns. The SHIFT LEFT command can be used only to reverse the effect of the SHIFT RIGHT command.
SHIFT RIGHT	Moves the current window to the right a specified number of columns, allowing you to view columns of characters that do not currently appear on the terminal screen.
SHRINK WINDOW	Shrinks the current window by a specified number of lines. For example, SHRINK WINDOW 5 shrinks the window by five lines. The adjacent window expands accordingly.
SPLIT WINDOW	Splits the current window, forming two smaller windows. You can divide the window into more than two parts by specifying a number with the command. For example, SPLIT WINDOW 3 splits the window into three windows.
TWO WINDOWS	Same as the SPLIT WINDOW 2 command.

1.8.1 Viewing One Buffer

To view two sections of a file at the same time, use the SPLIT WINDOW command. EVE splits your screen and creates two identical windows. The cursor maintains its position in the buffer but appears only in the bottom window. Notice that the buffer name is the same in both status lines.

Displaying two sections of a long file makes moving text within a file efficient. You can select and remove text from one part of the file and insert it into the other. To move the cursor from one window to the other, enter the NEXT WINDOW command.

To remove the second window from the screen and expand the current window to occupy the whole editing area, press the Do key, type the command ONE WINDOW, and press Return.

1.8.2 Editing Two Buffers

The following steps describe how to edit two buffers containing different files:

1 Create two windows on your screen by entering the SPLIT WINDOW command. EVE splits your screen and creates two windows. The cursor maintains its position in the buffer but appears only in the bottom window. Notice that the buffer name in each of the highlighted status lines is the same.

2 Put a second file in the current window using the GET FILE, OPEN, or OPEN SELECTED command.

To display in the current window a buffer that you created earlier in the editing session, enter the BUFFER command and the name of the buffer you want to display.

3 Your terminal screen now displays two different buffers. You can select and remove text from one buffer and insert it into the other buffer. To move the cursor from one window to the other, enter the command NEXT WINDOW.

The following example shows you how to edit two files and move text from one file to another using two windows. First, invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer: RHYMES.DAT                          | Write | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the Do key, type the command SPLIT WINDOW, and press Return to create two windows on your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer: RHYMES.DAT                          | Write | Insert | Forward
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer: RHYMES.DAT                          | Write | Insert | Forward
```

Press the Do key, type the command GET FILE SCHEDULE.DAT, and press Return to create a new buffer containing the text of SCHEDULE.DAT in the bottom window of your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]
Buffer: SCHEDULE.DAT                        | Write | Insert | Forward
4 lines read from file WORKDISK:[USER]SCHEDULE.DAT
```

While still in the bottom window, move the cursor to the letter *R* in the word *Read*. Press the Select key, then press the down arrow key twice. The last two lines in SCHEDULE.DAT are highlighted.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer: RHYMES.DAT                          | Write | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: SCHEDULE.DAT                        | Write | Insert | Forward
Move the text cursor to select text.
```

Press the Remove key to place the highlighted text in the Insert Here buffer.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer: SCHEDULE.DAT                        | Write | Insert | Forward
Remove completed.
```

Now enter the NEXT WINDOW command to move the cursor to the other window. Then, move the cursor to the bottom of buffer RHYMES.DAT and press the Insert Here key. The text that you removed from SCHEDULE.DAT is inserted into RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
Read and review memo from Sally
Work on Pascal program
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer: SCHEDULE.DAT                        | Write | Insert | Forward
```

Entering the ONE WINDOW command removes all other windows from the screen, expanding the window containing the cursor to occupy the whole editing area of the screen. Press the Do key, type ONE WINDOW, and press Return.

If you exit from the editing session, EVE writes the contents of the current buffer to a file and asks you if you want to write the contents of the other buffer to a file.

1.9 Defining Keys

You can define keys to execute EVE commands or to enter a series of keystrokes, called a **learn sequence**.

EVE does not let you define the Return key (Ctrl/M), the space bar, or any printing characters (such as letters, digits, and punctuation marks) on the main keyboard.

In addition, Digital recommends that you do not define the following keys and control key sequences (some cannot be defined unless you use special terminal settings):

Break
Delete or <⌫
Escape or Ctrl/[
F1 to F6 (VT200- and VT300-series terminals)
Help (PF2 on VT100-series terminals)
Noscroll
Shift
Ctrl/C
Ctrl/I (Tab key)
Ctrl/O
Ctrl/Q
Ctrl/R (which EVE defines as REMEMBER)
Ctrl/S
Ctrl/T
Ctrl/U (which EVE defines as START OF LINE)
Ctrl/X
Ctrl/Y

You can define all other keys, including control keys. You can redefine the Do key, as long as you have assigned the DO command to another key.

Note that the SET SHIFT KEY and SET NOSHIFT KEY commands are obsolete. Instead, use the SET GOLD KEY and SET NOGOLD KEY commands, respectively. For more information about setting a Gold key, see Section 1.9.3.

1.9.1 Defining Keys to Execute EVE Commands

Defining keys lets you create editing keys to enter EVE commands you use frequently. You can define a key to execute an EVE command by using the DEFINE KEY command or by using an initialization file. If you are using help and press a key to which you have assigned an EVE command, EVE provides the help text for that command. Key definitions are discarded when you end an EVE editing session, unless you use the SAVE ATTRIBUTES command or SAVE EXTENDED EVE command (see Section 1.12) to save key definitions from one editing session to the next.

The DEFINE KEY command assigns an EVE command to a single key, a Gold key combination, or control key sequence. You can enter the DEFINE KEY command, the key to be defined, and the command on a single command line or you can enter the DEFINE KEY command and let EVE prompt you.

To enter the DEFINE KEY command on a single command line, use the following command syntax:

```
DEFINE KEY [=key-name] command
```

The first parameter is the key to be defined; the second parameter is the command to assign to the key. For example, the following command assigns the MOVE BY WORD command to keypad key 1:

```
Command:  DEFINE KEY=KP1 MOVE BY WORD
```

The following command assigns the FILL command to Ctrl/F:

```
Command:  DEFINE KEY=Ctrl/F FILL
```

You can use one of three different separators when specifying key names: an underscore, a dash, or a slash. For example, the Ctrl/F key can be specified as Ctrl_F, Ctrl–F, or Ctrl/F.

The following example shows you how to use the DEFINE KEY command letting EVE prompt you. Invoke EVE and enter the DEFINE KEY command.

```
[End of file]

Buffer: MAIN                                  | Write | Insert | Forward
Command:  DEFINE KEY
```

Type the EVE command you want to assign to a key and press Return.

```
[End of file]

Buffer: MAIN                                  | Write | Insert | Forward
EVE command:  START OF LINE
```

Press the key to be associated with the EVE command.

```
[End of file]

Buffer: MAIN                                  | Write | Insert | Forward
Press the key you want to define:  F20
```

The message *Key defined* appears if you have successfully defined a key.

Another way to assign EVE commands to keys is to create an initialization file. An initialization file lets you define keys and set the characteristics of an editing session. The file contains EVE commands and key definitions and is executed when you invoke EVE. To define keys in an initialization file, put DEFINE KEY commands in the file using the syntax given in this section. For more information about initialization files, see the online help topic Initialization Files.

To remove a key definition, use the UNDEFINE KEY command.

Section 1.9.3 contains more examples of defining keys to execute EVE commands.

1.9.2 Defining Keys to Enter Learn Sequences

The LEARN command assigns a sequence of keystrokes, called a **learn sequence**, to a single key or control key sequence. Learn sequences allow you to enter the same series of keystrokes in a buffer any number of times, simply by pressing one key. All learn sequences are discarded when you terminate an EVE editing session unless you use the SAVE ATTRIBUTES command or the SAVE EXTENDED EVE command (see Section 1.12) to save them from one editing session to the next.

Define a learn sequence as follows:

1. Enter the LEARN command.
2. Type the keystrokes to be remembered. You can press keys already defined, or type text, or both.
3. Press Ctrl/R. At the prompt, press the key to be associated with the learn sequence, such as F17 or PF3. To cancel the learn sequence, press Return or Ctrl/M.

The message *Key sequence remembered* appears if you have successfully defined a key.

The following example shows you how to define a learn sequence that inserts a string of text into your file when you press Ctrl/F. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

First, move the cursor to the end of the buffer. To begin the definition of the learn sequence, enter the LEARN command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  LEARN
```

Insert the following text, which EVE is to remember, at the end of your file: *And what is a rhyme?*

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Press keystrokes to be learned. Press Ctrl/R to remember these keystrokes.
```

Press Ctrl/R.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Press the key that you want to use to do what was just learned:
```

Press Ctrl/F, the key to which you are assigning the learn sequence.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
```

```
Buffer: RHYMES.DAT                          | Write | Insert | Forward
Key sequence remembered.
```

For the rest of the editing session, press Ctrl/F and EVE inserts the text *And what is a rhyme?* wherever the cursor is positioned at the time.

1.9.3 Defining a Gold Key

You can assign two definitions to the same editing key if you create a Gold key. Invoke one key definition by pressing the editing key. Invoke the other key definition by first pressing the Gold key and then pressing the editing key.

To define a Gold key, enter the SET GOLD KEY command and press the key you want to use as the Gold key. When you successfully define the key, the message *Gold key set.* appears in the Messages buffer. EVE does not have a default Gold key.

After you create a Gold key, you can use the Gold key definitions supplied by EVE. To see a diagram of these key definitions, enter the command HELP KEYPAD. The Gold key definitions appear in the display in reverse video.

The following table lists the Gold key combinations on the EVE keypad and the definitions associated with them:

Key	Definition
Gold–F13	Restore Word (except with the WPS keypad)
Gold–Help	Help Keys
Gold–Find	Wildcard Find
Gold–Insert Here	Restore
Gold–Remove	Store Text
Gold–Select	Reset
Gold–Prev Screen	Previous Window
Gold–Next Screen	Next Window
Gold–↑	Top
Gold–↓	Bottom
Gold–←	Start of Line
Gold–→	End of Line

You also can create your own key definitions using the Gold key. The following example demonstrates how to define a Gold key and assign two commands to a single key. The example defines the 4 key on the numeric keypad as the Gold key and then assigns the BOTTOM and TOP commands to the Ctrl/G key. Thus, pressing Ctrl/G alone enters the BOTTOM command, and pressing the Gold key followed by Ctrl/G enters the TOP command.

Define a Gold key by pressing the Do key, typing SET GOLD KEY, and pressing Return.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer:  RHYMES.DAT                    | Write | Insert | Forward
Command:  SET GOLD KEY
```

Press the 4 key on the numeric keypad.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                     | Write | Insert | Forward
Press the key you want to use as Gold:  KP4
```

Press the Do key, and enter the DEFINE KEY command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  DEFINE KEY
```

Enter the BOTTOM command.

```
And what is a rhyme?
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
EVE command:  BOTTOM
```

Press Ctrl/G.

```
And what is a rhyme?
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Press the key you want to define:  Ctrl/G
```

Ctrl/G is now defined as BOTTOM.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Key defined.
```

Now define the Gold–Ctrl/G key as the TOP command. Press the Do key, and enter the DEFINE KEY command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Command:  DEFINE KEY
```

Enter the TOP command.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
EVE command:  TOP
```

To assign a key for the TOP command, press and hold down the Gold key (4 on the numeric keypad) and press Ctrl/G.

```
And what is a rhyme?
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer: RHYMES.DAT                          | Write | Insert | Forward
Press the key you want to define:  KP4 Ctrl/G
```

For the rest of your editing session, when you press Ctrl/G, EVE executes the BOTTOM command, and when you press the Gold key (4 on the numeric keypad) followed by Ctrl/G, EVE executes the TOP command. If you press the Gold key by mistake, press the Select key to cancel it. Use the SAVE ATTRIBUTES command or the SAVE EXTENDED EVE command (see Section 1.12) to save key definitions from one editing session to the next.

You cannot define more than one Gold key at a time. To remove a Gold key definition, enter the SET NOGOLD KEY command, then press the key you want to undefine. Or, define another Gold key, which removes the original Gold key.

Another way of defining a Gold key is to insert a command in an initialization file, using the following format:

```
SET GOLD KEY  keyname
```

For example, the following command defines the PF1 key as the Gold key:

```
SET GOLD KEY PF1
```

1.10 Using DCL Within EVE

You can execute a DCL command from within EVE, or you can use a subprocess to switch between the DCL command level and an EVE editing session quickly.

1.10.1 Executing a DCL Command

To enter a DCL command from within EVE, enter the EVE command DCL with the DCL command you want to execute, and press Return. The message *Creating DCL subprocess . . .* appears in the message window.

```
[End of file]

Buffer: MAIN                                | Write | Insert | Forward
Command:  DCL SHOW PROCESS
```

After the DCL command has executed, EVE creates another window, if necessary, and displays the DCL command and its output in the DCL buffer. (The cursor remains in the buffer it was in before you executed the DCL command.)

```
[End of file]

Buffer: MAIN                                | Write | Insert | Forward
SHOW PROCESS

19-APR-1990 11:42:49.63   User: SMITH              Process ID:   1234567A
                          Node: MYVAX              Process name: "SMITH_1"

Terminal:
User Identifier:    [001,SMITH]
Base priority:      4
Default file spec:  USER$:[SMITH]

Buffer: DCL                    | Read-only | Insert | Forward
Subprocess activated
```

You can move the cursor to the DCL buffer using the NEXT WINDOW command, select and remove text, and copy it to the editing buffer. Do not enter DCL commands that generate continuous output or run programs that do screen management of their own, such as the Phone Utility.

Some DCL commands continue to control the subprocess for DCL commands that follow. For example, enter the EVE command DCL HELP DIRECTORY to spawn a subprocess that gives you the DCL HELP entry on DIRECTORY.

```
Command:  DCL HELP DIRECTORY
```

The DCL HELP DIRECTORY is displayed in the DCL buffer.

```
[End of file]

Buffer: MAIN                                | Write | Insert | Forward
HELP DIRECTORY

DIRECTORY

    Provides a list of files or information about a file or group of
    files.

    Format:

      DIRECTORY  [file-spec[,...]]

  Additional information available:

  Parameters Command_Qualifiers
  /ACL       /BACKUP     /BEFORE     /BRIEF      /BY_OWNER   /COLUMNS
  /CREATED   /DATE       /EXCLUDE    /EXPIRED    /FILE_ID    /FULL
  /GRAND_TOTAL           /HEADING    /MODIFIED   /OUTPUT     /OWNER
  /PRINTER   /PROTECTION             /SECURITY   /SELECT     /SINCE
  /SIZE      /TOTAL      /TRAILING   /VERSIONS   /WIDTH
  Examples

Buffer: DCL                    | Read-only | Insert | Forward
Subprocess activated
```

Then enter the EVE DCL command SHOW PROCESS.

```
Command:  DCL SHOW PROCESS
```

The subprocess treats that DCL command as a request for help on the topic DIRECTORY SHOW PROCESS.

```
[End of file]

Buffer: MAIN                          | Write | Insert | Forward

SHOW PROCESS

DIRECTORY
  Sorry, no documentation on DIRECTORY SHOW PROCESS

  Additional information available:

  Parameters Command_Qualifiers
  /ACL       /BACKUP     /BEFORE    /BRIEF     /BY_OWNER  /COLUMNS
  /CREATED   /DATE       /EXCLUDE   /EXPIRED   /FILE_ID   /FULL
  /GRAND_TOTAL           /HEADING   /MODIFIED  /OUTPUT    /OWNER
  /PRINTER   /PROTECTION            /SECURITY  /SELECT    /SINCE
  /SIZE      /TOTAL      /TRAILING  /VERSIONS  /WIDTH
  Examples

Buffer: DCL                           | Read-only | Insert | Forward

Subprocess activated
```

To exit from DCL Help, enter the following commands:

```
Command:  DCL
DCL Command:  Ctrl/V
Press the key to be added:  Ctrl/Z
```

For more information about using the DCL command, see the online help topic DCL.

1.10.2 Creating a Subprocess

You can create a subprocess to switch between an EVE editing session and DCL command level without terminating your editing session. To create a subprocess, enter the SPAWN command. EVE suspends the current editing session and connects your terminal to a new VMS subprocess. The DCL prompt ($) appears on your terminal screen.

Note: The SPAWN and ATTACH commands are not supported if you invoke EVE using the /INTERFACE=DECWINDOWS qualifier..

The most common reasons to spawn a subprocess are to invoke the Mail Utility (MAIL) and to run screen-oriented programs, although your subprocess can invoke any VMS utility or execute any DCL command.

To return to your editing session, log out of the subprocess by entering the DCL command LOGOUT. EVE resumes the editing session, and the cursor appears in the location it occupied before you spawned the subprocess.

Alternatively, you can supply a DCL command as a parameter to the SPAWN command to create a specific subprocess. The following example executes the Mail Utility. Enter the SPAWN MAIL command.

```
[End of file]

Buffer: MAIN                                    | Write | Insert | Forward
Command:  SPAWN MAIL
```

The prompt for the Mail Utility (MAIL>) appears on the screen. When you exit from MAIL, you are automatically logged out of the subprocess and EVE resumes the editing session.

```
MAIL>  EXIT
```

Rather than spawn a process to use DCL, you can spawn a process for an EVE editing session and then attach to the parent DCL process to use DCL commands and utilities.

First, create a subprocess using the DCL command SPAWN.

```
$  SPAWN
%DCL-S-SPAWNED, process SMITH_1 spawned
%DCL-S-ATTACHED, terminal now attached to process SMITH_1
```

The SPAWN command creates a subprocess (displayed by the SHOW PROCESS command as *SMITH_1*). At the subprocess level, invoke EVE and conduct the editing session.

```
$  EDIT/TPU
```

When you want to return to the DCL command level, use the EVE command ATTACH to return to the parent process (SHOW PROCESS displays the process as *SMITH*).

```
[End of file]

Buffer: MAIN                                    | Write | Insert | Forward
Command:  ATTACH SMITH
```

To resume your editing session, reconnect to the editing subprocess using the DCL command ATTACH with the process name of the subprocess (*SMITH_1*). EVE resumes the editing session and the cursor appears in the location it occupied before you attached to the parent process.

```
$  ATTACH SMITH_1
```

1.11 Using VAXTPU Procedures to Extend EVE

EVE is an editor built on the VAX Text Processing Utility (VAXTPU), which is a programmable text processor. The EVE command TPU allows you to enter any VAXTPU statement or series of statements that can be expressed on one command line.

To enter a TPU command, enter the command TPU followed by the VAXTPU statement you want to execute. For example, to execute the VAXTPU statement APPEND_LINE, which places the current line at the end of the previous line, enter the following command string:

```
Command:  TPU APPEND_LINE
```

For more information about the TPU command, type HELP TPU. See the *VAX Text Processing Utility Manual* for a complete list of VAXTPU statements and procedures.

1.11.1 Writing VAXTPU Procedures

Because EVE is an editor written in the VAXTPU programming language, you can extend EVE functions by writing procedures in VAXTPU. This section assumes that you are familiar with the VAXTPU programming language described in the *VAX Text Processing Utility Manual.*

Before you begin writing VAXTPU procedures to modify EVE, Digital recommends that you study the EVE source code, which is stored in SYS$EXAMPLE:EVE$*.TPU. (The wildcard character in the file specification indicates that EVE source code is stored in many files.) These files are put together using EVE$BUILD. Variables and statements in your procedures should be consistent with EVE variables and statements so that you can avoid making changes that adversely affect EVE operations.

You can write procedures in the VAXTPU programming language that are, in effect, new EVE commands. When you write new EVE command procedures, you should follow these rules:

- Prefix the procedure name with the label *EVE_* so that EVE recognizes the procedure as an EVE command. After you compile the procedure, execute it by pressing the Do key and typing the procedure name without the EVE_ prefix or the underscores. For example, enter SET LEFT MARGIN for the EVE_SET_LEFT_MARGIN procedure. (You can also define a key to execute the new command. See Section 1.9.)
- The words *PROCEDURE* and *ENDPROCEDURE* must start in column 1.
- Initialize all global variables in a single procedure named TPU$LOCAL_INIT or in a module initialization procedure if you build with EVE$BUILD.

If the EVE command you are writing takes any integer parameters, such as SET LEFT MARGIN 2, where 2 is the parameter, you must define a global variable to associate that parameter with the data-type integer. When you are using the EVE commands that you have defined, EVE passes the null string ("") if you do not provide a value for a string parameter; it passes the value EVE$K_NO_ARG if you do not provide a value for an integer parameter.

Digital recommends placing global variables in a procedure called TPU$LOCAL_INIT, which is called each time EVE starts. Place all procedures that use parameters in the same file that contains the TPU$LOCAL_INIT procedure. The following is an example of a global variable definition for command parameters:

- The following definition of the global variable EVE$ARG1_ADD tells EVE to expect an integer as the first parameter for EVE_ADD:

```
eve$arg1_add := "integer";
```

- The following definition of the global variable EVE$ARG1_HELLO tells EVE to expect a string as the first parameter to EVE_HELLO:

```
eve$arg1_hello := "string";
```

The integer variables are required.

In general, each of the global variable names that define command parameters must consist of the following two parts:

- The first part of the variable name must be *eve$*.
- The second part defines the variable's sequence within a procedure. For the first variable, it is *arg1*; for the second, *arg2*; and so on.
- The third part of the variable name is the procedure name without the *EVE_* prefix.

1.11.2 Compiling VAXTPU Procedures

The EXTEND EVE command enables you to compile a VAXTPU procedure without leaving EVE. To compile the procedure that the cursor is in, use the EXTEND THIS command. To compile one procedure, enter the command EXTEND EVE and the name of the procedure you want to compile. To compile all procedures in a file, enter the command EXTEND EVE * (which is the same as the EVE command EXTEND ALL). If you miss a message from the compiler, use the command BUFFER MESSAGES to read the messages stored in the Messages buffer.

The following example illustrates how to create and compile VAXTPU procedures to define an ADD command, a HELLO command, and the parameters for both commands. Invoke EVE to edit the file MYPROCEDURES.TPU and insert the following text into the file:

```
! Procedure to add two integers and display the result in the message window

procedure eve_add (a1, a2)

local   temp,
        n1,
        n2;

if not eve$prompt_number (a1, n1, "First number to add: ",
                          "No number specified.")
then
    return (FALSE);
endif;

if not eve$prompt_number (a2, n2, "Second number to add: ",
                          "No number specified.")
then
    return (FALSE);
endif;

temp := n1 + n2;
message (str (n1) + " + " + str (n2) + " = " + str (temp));
return (TRUE);

endprocedure;


procedure eve_hello (my_name)
local   the_name;

if eve$prompt_string (my_name, the_name, "Name: ", "We haven't been introduced")
then
    message ("Hello " + the_name);
    return (TRUE);
else
    return (FALSE);
endif;
```

```
endprocedure;

eve$arg1_add := "integer";
eve$arg2_add := "integer";
Buffer: MYPROCEDURES.TPU                    | Write | Insert | Forward
```

Put the definitions for the two parameter variables in your TPU$LOCAL_INIT procedure, using the syntax shown in the file MYPROCEDURES.TPU.

To compile the procedures you have entered into MYPROCEDURES.TPU, press the Do key, type EXTEND EVE *, and press Return. If you are going to use the newly compiled commands in the current editing session, you must execute TPU$LOCAL_INIT by entering the command TPU TPU$LOCAL_INIT.

1.12 Saving Key Definitions, Attributes, and VAXTPU Procedures

You customize EVE to meet your editing requirements in the following ways:

- Defining keys

  You can assign an editing command to a key for entering a command quickly.

- Creating learn sequences

  You can assign a series of commands or keystrokes to one key. For example, you can create a learn sequence where you press one key to insert a change of address into a standard memo heading.

- Setting attributes

  Some of the attributes you set are global settings, applying to the editor or to all buffers; others are buffer-specific, applying to particular buffers. For example, you can set global attributes such as tab modes and wildcard patterns and buffer-specific attributes such as margins and tab stops.

- Writing editing procedures using VAXTPU

  You can expand EVE beyond its standard set of commands by using VAXTPU procedures. With VAXTPU language statements, you can write procedures that create features that are not available in EVE. For example, you can write a procedure to transpose two characters and assign this procedure to a key.

Section 1.9 describes how to define keys for commands and for learn sequences. You can change settings, or attributes, when editing text or formatting a buffer, as described in Section 1.3 and Section 1.6. Section 1.11 describes how to write VAXTPU procedures.

You can save your modifications to EVE for your future editing sessons in startup files. Startup files hold key definitions and editing commands that set the characteristics of the editing environment. Startup files can also hold VAXTPU procedures, which augment the editing capability of the standard EVE editor. EVE has three types of startup files:

- Section file

  A section file contains key definitions, learn sequences, and compiled VAXTPU statements and procedures in binary form. Because section files are in binary form, they set up the editing environment very quickly, but you cannot display or edit a binary file. Use a section file to implement editing features that are not likely to change from one editing session to another.

- Command file

  A command file is a VAXTPU source file containing VAXTPU statements and procedures. You can use a command file in two different ways. A command file can be used to generate an EVE section file. Or, it can be used as a command file that VAXTPU executes to create an editing environment. For example, you can use a command file to control the appearance of the buffer and the startup mode of the editor. Command files execute more slowly than section files, but because they are ASCII files, you can display and edit them.

- Initialization file

  An initialization file is an ASCII file containing EVE commands. You can easily display an initialization file and edit it. You can execute an initialization file when invoking EVE, or during an editing session with the @ command. Initialization files set up the editing environment more slowly than section files or command files. They can contain key definitions but not learn sequences.

During your editing session, you can save key definitions (including learn sequences), some attributes, menu definitions, compiled VAXTPU procedures, and other extensions to the editor in a section file or command file by using the SAVE ATTRIBUTES command or the SAVE EXTENDED EVE command, or as part of exiting or quitting.

Table 1–3 summarizes the commands for saving attributes.

Table 1–3 EVE Commands for Saving Attributes

Command	Usage or Effects
SAVE ATTRIBUTES	Saves attribute settings and user menu definitions in a section file or command file, depending on your responses to EVE prompts or on settings done with other EVE commands. If you save in a section file, the effect is the same as SAVE EXTENDED EVE. If you save in a command file, EVE generates a specially marked block of VAXTPU statements for attribute settings and menu definitions, and either creates a command file or updates an existing command file with this block of statments.
SAVE SYSTEM ATTRIBUTES	Saves EVE default attribute settings in a section file or command file. This is useful if you want to restore your section file or command file to the standard EVE settings and menu definitions.

(continued on next page)

Table 1–3 (Cont.) EVE Commands for Saving Attributes

Command	Usage or Effects
SAVE EXTENDED EVE	Creates a section file, saving attribute settings, key definitions, menu definitions, compiled procedures, and other extensions, such as global variables set with a VAXTPU statement. If you do not specify a section file on the command line, EVE prompts you for one or uses your default section file (if you set a default).
SET DEFAULT COMMAND FILE	Determines the command file for saving attributes. Does not determine the command file to be executed at startup, if any.
SET DEFAULT SECTION FILE	Determines the section file for saving attributes. Does not determine the section file to be executed at startup.
SET EXIT ATTRIBUTE CHECK	Default setting. If you changed attributes, then when you exit or quit, EVE asks if you want to save your changes.
SET NODEFAULT COMMAND FILE	Default setting. When you save attributes, the default command file is TPU$COMMAND.TPU in your current directory, or the command file that was executed at startup.
SET NODEFAULT SECTION FILE	Default setting. When you save attributes, EVE asks for the name of the section file you want to create (unless you disabled section file prompting).
SET NOEXIT ATTRIBUTE CHECK	Disables attribute checking, typically to speed up or simplify exiting or quitting. Does not apply to the editing session in which you issue the command, but only to the editing sessions in which you use the saved section file or command file.
SET NOSECTION FILE PROMPTING	Disables prompting for a section file when you save attributes, typically to speed up or simplify saving attributes in a default section file or in a command file.
SET SECTION FILE PROMPTING	Default setting. When you save attributes, EVE prompts you for the name of a section file.

By default, if you have changed attributes and not saved them, then on exiting EVE prompts you as in the following example:

```
Command:  SET CURSOR BOUND
Command:  MOVING_TEXT
Command:  SET TABS VISIBLE
            .
            .
            .
Command:  EXIT
Attributes were changed.  Save them? [YES]
```

If you want to save the changes, simply press Return. Effectively, EVE then does a SAVE ATTRIBUTES command before exiting. If you do not want to save the changes, type NO and press Return. EVE then continues exiting.

To disable this prompting, to make exiting faster or simpler, use the SET NOEXIT ATTRIBUTE CHECK command. However, the command does not apply to the current editing session because exit checking is itself a global setting and can be saved in a section file or command file. After you save it, the setting applies to future editing sessions in which you use the relevant section file or command file.

Other global attributes (such as scroll margins or the type of wildcards) and any buffer-specific attributes (such as margins or tab stops) are not saved in a section file or a command file. Typically, you use an initialization file for those settings.

There are several ways in which you can use all your customizations in future editing sessions. You can combine different types of startup files in the following ways:

- Section file and an initialization file
- Command file (EVE-generated code) and an initialization file

  If you have limited disk space, use a command file instead of a section file.
- Command file (EVE-generated and user-generated code)

  You can write VAXTPU procedures for attributes that typically are contained in an initialization file and use them in a command file containing EVE-generated code. Command files execute more quickly than an initialization file and offer more sophisticated editing tools.

Table 1-4 lists the categories of commands for EVE attributes and features and shows in what type of startup file you can save them.

Table 1–4 EVE Startup Files

Attribute/Feature	Section	Command	Initialization
Key Definitions			
DEFINE KEY	X	X	X
LEARN	X		
SET FUNC KEYS [NO]DECWINDOWS	X	X	X
SET [NO]GOLD KEY	X	X	X
SET KEYPAD [NO]EDT	X	X	X
SET KEYPAD [NO]WPS	X	X	X
SET KEYPAD VT100	X	X	X
SET KEYPAD NUMERIC	X	X	X
UNDEFINE KEY	X	X	X

(continued on next page)

Table 1–4 (Cont.) EVE Startup Files

Attribute/Feature	Section	Command	Initialization
Global Settings–1			
SET BOX [NO]PAD	X	X	X
SET BOX [NO]SELECT	X	X	X
SET CURSOR FREE or BOUND	X	X	X
SET [NO]CLIPBOARD	X	X	X
SET [NO]DEFAULT COMMAND FILE	X	X	X
SET [NO]DEFAULT SECTION FILE	X	X	X
SET [NO]EXIT ATTRIBUTE CHECK	X	X	X
SET FIND CASE [NO]EXACT	X	X	X
SET [NO]PENDING DELETE	X	X	X
SET [NO]SECTION FILE PROMPTING	X	X	X
SET TABS INSERT, MOVEMENT, or SPACES	X	X	X
SET TABS [IN]VISIBLE	X	X	X
Global Settings–2			
SET FIND [NO]WHITESPACE			X
SET SCROLL MARGINS			X
SET WIDTH			X
SET WILDCARDS VMS or ULTRIX			X
Buffer Settings			
FORWARD or REVERSE			X
INSERT MODE or OVERSTRIKE MODE			X
SET BUFFER			X
SET [NO]JOURNALING ALL			X
SET LEFT MARGIN			X
SET PARAGRAPH INDENT			X
SET RIGHT MARGIN			X
SET TABS AT or EVERY			X
SET [NO]WRAP			X
VAXTPU Procedures			
	X	X	

By placing your definitions and procedures in a startup file, you can invoke the editor and automatically establish the editing environment your task requires. Section 1.12.1, Section 1.12.2, and Section 1.12.3 describe how to create and use section files, command files, and initialization files.

1.12.1 Saving in a Section File

Typically, you save customized settings in a section file. A section file is in binary form and saves attributes, key definitions (including learn sequences), menu definitions, compiled procedures, and other extensions to the editor, including any saved in the section file you are using. Because the section file is binary, it is executed quickly at startup. The default file type for section files is TPU$SECTION.

To create a section file, you can use the SAVE EXTENDED EVE command or the SAVE ATTRIBUTES command. Using SAVE EXTENDED EVE, you can specify the section file on the command line or let EVE prompt you for the section file name. Using SAVE ATTRIBUTES, you specify the section file as a response to a prompt.

For example, the following command saves attributes and other customized settings in a section file called MYSEC.TPU$SECTION in your current directory:

```
Command:  SAVE ATTRIBUTES
Save attributes in a section file [YES]? Return
File to save in:  mysec
DISK$1:[USER]MYSEC.TPU$SECTION;1 created
```

To speed up saving in a section file, you can set a default section file—that is, the section file you want to save in without having to specify the file each time you save attributes—and you can disable section file prompting. Table 1–5 shows the interaction of the settings for default section file and section file prompting.

Table 1–5 EVE Settings for Saving Attributes

Commands (Settings)	Effects with SAVE ATTRIBUTES
SET DEFAULT SECTION FILE SET SECTION FILE PROMPTING	When you save attributes, EVE ask you whether to save in a section file. If you respond YES (the default response), EVE saves in your default section file. If you respond NO, EVE asks whether to save in a command file.
SET DEFAULT SECTION FILE SET NOSECTION FILE PROMPTNG	When you save attributes, EVE saves in your default section file without prompting.
SET NODEFAULT SECTION FILE SET SECTION FILE PROMPTING	Default settings. When you save attributes, EVE asks whether to save in a section file. If you respond YES, EVE asks for the name of a section file. If you respond NO, EVE asks whether to save in a command file.
SET NODEFAULT SECTION FILE SET NOSECTION FILE PROMPTNG	When you save attributes, EVE asks whether to save in a command file.

Typically, when you use SET DEFAULT SECTION FILE, you specify the section file you are going to use at startup for future editing sessions. The command does not determine the section file to be executed when you invoke the editor, but only the section file in which you save attributes and other customized settings. To specify the section file you want executed at startup, do either of the following:

- Use EDIT/TPU/SECTION and specify the section file you want to use.

```
$ EDIT/TPU/SECTION=myeve
```

- Define the logical name TPU$SECTION to specify the section file, and and then use the EDIT/TPU command.

```
$ DEFINE TPU$SECTION sys$login:myeve
$ EDIT/TPU
```

Note that in specifying the section file to be executed, you must use a complete file specification, including the device (or disk) and directory. Otherwise, VAXTPU assumes the section file is in SYS$SHARE.

Section files may be quite large, depending on the number of customized settings you save. If you have limited disk space, you should save in a command file, which requires less disk space. For more information about creating and using section files, see the EVE online help topic called Section Files.

1.12.2 Saving in a Command File

A command file contains attribute settings and also VAXTPU procedures and statements that are compiled and executed at startup—in effect, a series of programs for extending EVE. A command file may be slower at startup than a section file (depending on the number of procedures to be compiled and statements to be executed), but it takes up less disk space than a section file, and a command file can be edited and printed. Also, if you edit your command file, you can recompile procedures during your editing session by using EXTEND commands. The default file type for command files is TPU.

When you use the SAVE ATTRIBUTES command, or when you save attributes on exiting or quitting, you can have EVE create or update a command file. To save attributes in a command file, use the SAVE ATTRIBUTES command, as follows:

```
Command:  SAVE ATTRIBUTES
Save attributes in a section file [YES]?  no
Save attributes in a command file [YES]?  Return
Enter file name [TPU$COMMAND.TPU]  mycom
14 written to file DISK$1:[USER]MYCOM.TPU;1
```

Note that the prompt for the command file name shows, in brackets, the default command file that EVE uses if you simply press Return at the prompt without typing a file name. This default is one of the following:

- The command file specified with the /COMMAND qualifier when you invoked EVE
- The command file defined by the logical name TPU$COMMAND
- A command file called TPU$COMMAND.TPU in your current directory

You can set your preferred default command file—that is, the command file you want EVE to create or update, without having to specify the file each time you save attributes. For example, the following command sets your default command file as MYCOM.TPU in your current directory:

```
Command:  SET DEFAULT COMMAND FILE mycom
```

If you want to save in a command file rather than in a section file, you should also use the SET NOSECTION FILE PROMPTING command. Then, when you save attributes, EVE asks whether to save in a command file without first asking whether to save in a section file.

Typically, when you use SET DEFAULT COMMAND FILE, you specify the command file you are going to use at startup for future editing sessions. The command does not determine the command file to be executed when you invoke EVE, but only the command file in which you save attributes and menu definitions. To specify the command file you want executed at startup, do any of the following:

- Use EDIT/TPU/COMMAND and specify the command file you want to use.

```
$ EDIT/TPU/COMMAND=myeve
```

- Define the logical name TPU$COMMAND to specify the command file, and then invoke EVE using EDIT/TPU.

```
$ DEFINE TPU$COMMAND sys$login:myeve
$ EDIT/TPU
```

- Name the command file TPU$COMMAND.TPU in your current directory, and then invoke EVE using EDIT/TPU.

For more information about creating and using command files, see the EVE online help topic called Command Files.

1.12.3 Using an Initialization File

Rather than defining keys or setting the characteristics of an editing session interactively, you can put EVE commands and key definitions in an initialization file. You can execute an initialization file when invoking EVE or during an editing session by using the @ command. For example:

```
Command:  @SETUP_INIT
```

Each command in an initialization file begins on a separate line. You can add comments to the file to document it, as long as you precede the comments with an exclamation point and place them on a line separate from a command. An initialization file has a file type of EVE.

The following is an example of an initialization file:

```
SET LEFT MARGIN 4
SET PARAGRAPH INDENT +3
SET RIGHT MARGIN 72
SET TABS EVERY 10
SET SCROLL MARGINS 9% 9%
SET FIND WHITESPACE
!Key definitions
SET KEYPAD EDT
DEF KEY= F20  SHOW BUFFERS
DEF KEY= Ctrl/P  BOX PASTE
DEF KEY= Gold-S  BOX SELECT
DEF KEY= KP7  WPS Gold-R
```

An initialization file can be specified with the /INITIALIZATION qualifier, defined as EVE$INIT in your LOGIN.COM file, or named EVE$INIT.EVE in your SYS$LOGIN directory. The following command invokes EVE with the initialization file named MY_INIT:

```
$ EDIT/TPU/INIT=WORK1:[ALEXIS]MY_INIT
```

By default, VAXTPU uses the initialization file whose logical name is EVE$INIT. If you define this logical name in your LOGIN.COM file, VAXTPU automatically uses your initialization file when you invoke EVE. For example, you could insert the following command in your LOGIN.COM file:

```
$ DEFINE EVE$INIT WORK1:[ALEXIS]MY_INIT.EVE
```

When EVE starts up, it looks first for a section file, then for a command file, and finally for an initialization file. Since an initialization file is executed after a section file and a command file, the definitions in an initialization file override those in a section file or a command file. For this reason, place commands that define the editing environment in either your command file or your initialization file.

2 Editing Files with EDT

EDT is an interactive text editor. You can use EDT to edit many kinds of text files—letters, memos, or complex computer programs. With EDT you can create new files, insert text into them, and edit that text. You can also edit text in existing files.

2.1 Introduction

EDT offers many features to make text editing easier and more efficient. These features include the following:

- Three types of editing—keypad mode, line mode, and nokeypad mode. Keypad mode is screen-oriented, which allows you to see several lines of text simultaneously and move the cursor throughout the text in any direction. Line mode enables you to edit text by using line numbers. You can use nokeypad mode commands to define keys. You can move from one mode to another during the same editing session.
- Online help. You can use help any time during your editing session without affecting your work.
- Journal facility. The journal file protects your editing work in case of a system interruption.
- Access to files and buffers. You can work with as many files and buffers as you need during your EDT session.
- Startup command files. These enable you to personalize the characteristics of your editing sessions.
- Key definition facility. You can define keys to automate your keypad editing work.
- EDT macros. You can create EDT macros to automate your editing work. Macros can be saved for use in later editing sessions.
- Tabbing facility. This feature enables you to create layered text formats.

The discussion of EDT in this guide is designed for the novice as well as the more experienced EDT user. For more detailed information about the EDT editor, see the *VAX EDT Reference Manual*.

2.1.1 Invoking and Terminating EDT

An editing session begins when you invoke EDT using the DCL command EDIT and ends when you terminate EDT with the EXIT or QUIT command. You can start an editing session by creating a file and inserting the text for the file during the course of the session, or you can specify an existing file when you start the session. EDT does not destroy the contents

of any existing file that you edit; it simply produces a new version, leaving the old version intact.

2.1.1.1 Invoking EDT

To invoke EDT, enter the DCL command EDIT. You are prompted for the name of a file. The following example shows how you invoke EDT to edit a file named MEMO.TXT:

```
$ EDIT
_File: MEMO.TXT
```

If you are creating a new file, EDT displays the message *Input file does not exist* followed by the end-of-buffer sign [EOB] and the asterisk prompt (*). If you are editing an existing file, a copy of the first line of text appears on the screen followed by the asterisk prompt.

The following example demonstrates how to invoke EDT to create a new file:

```
$ EDIT NEWFILE.FUN
Input file does not exist
[EOB]
*
```

The following example demonstrates how to invoke EDT to edit a file that already exists:

```
$ EDIT OLDFILE.DAT
    1       This is the first line of the file.
*
```

Once you enter EDT, you can choose between three different modes:

- Line mode—Line mode allows you to use line numbers during your editing session. Line-mode commands are particularly useful for manipulating large blocks of text.
- Keypad mode—Keypad mode allows you to move the cursor to the text you want to edit and perform most editing functions by pressing keypad keys. In keypad mode, you insert text by typing from the main keyboard.
- Nokeypad mode—Nokeypad mode allows you to move the cursor directly to the text you want to modify and to enter nokeypad commands.

One difference between the three modes is that you press keys to edit text in keypad mode, whereas you enter commands to edit text in line mode and in nokeypad mode.

By default, EDT puts you into line mode. You know that you are in line mode when you see the asterisk prompt (*). If you want to enter keypad mode, type the letter *C* (abbreviation for CHANGE command) when you see the asterisk prompt and press Return. If you want to enter line mode from keypad mode, press Ctrl/Z. If you want to enter nokeypad mode, enter the SET NOKEYPAD command when you see the asterisk prompt, press Return, type the letter *C*, and press Return again.

The following table shows the three EDT modes and the commands you enter to invoke each mode on a video terminal.

Mode	Commands to Invoke Mode
Line	$ EDIT *
Keypad	$ EDIT * C
Nokeypad	$ EDIT * SET NOKEY * C

2.1.1.2 Terminating EDT

Both the EXIT and QUIT commands terminate an editing session; however, only EXIT saves your edits. When you enter the EXIT command at the asterisk prompt, EDT creates an output file containing the edited version of the input file. By default, the output file has the same name and type as the input file. (The version number is incremented by one.) The following example demonstrates how to invoke EDT to edit a file named FUN.DAT and how to terminate the editing session. The output file has the same name as the input file and the version number is increased by one.

```
$  EDIT FUN.DAT
    .
    .
    .
  Ctrl/Z
* EXIT
DBA2:[WHITEBRIDGE]FUN.DAT;5  2 lines
```

If you want to override the default and specify a different output file name, enter the EXIT command and specify the new file name as a parameter. In the following example, you invoke EDT to edit a file named FUN.DAT and specify JOKE.DAT as the new file name when you terminate the session.

```
$  EDIT FUN.DAT
    .
    .
    .
* EXIT JOKE.DAT
DBA2:[WHITEBRIDGE]JOKE.DAT;1  2 lines
```

If you do not want to save your edits when you end an editing session, enter the QUIT command. All the edits you made to the file are lost and no output file is created.

The following table shows the three EDT modes and the commands you enter to terminate each mode:

Editing Mode	Commands to Terminate EDT
Line	* EXIT or QUIT $
Keypad	Ctrl/Z * EXIT or QUIT $ or Gold key, Command key, EXIT or QUIT command, Enter key $
Nokeypad	EX * EXIT or QUIT $ or QUIT $

2.1.2 Using the Help System

EDT's online help system allows you to get help during your editing session without interrupting your work.

In keypad mode, press the Help key to get information about keypad functions. For information about line and nokeypad commands, use the line-mode command HELP. The following sections discuss the help system in more detail.

2.1.2.1 Accessing Help from Line Mode

You can enter the HELP command after the asterisk prompt (*) for information about EDT commands. If you want information about specific commands, type the HELP command followed by the name of the command and press Return. For example, if you want to know more about the INSERT command, enter the following:

```
* HELP INSERT
```

If you want information about a qualifier, for example/SAVE, enter the following:

```
* HELP EXIT/SAVE
```

You must include the name of the command, in this case EXIT, and the slash (/) before the qualifier.

EDT can help you get information even when you supply only the first letter or so of the command name. If you enter HELP D, EDT prints the information for both the DEFINE and DELETE commands. HELP CH

gives you the CHANGE command, but HELP C supplies information for CHANGE, CLEAR, and COPY.

Use the wildcard character (*) with the HELP command to get information about all subtopics for a command without having to supply the subtopic names. For example, the following command prints information about TYPE/BRIEF and TYPE/STAY:

```
* HELP TYPE *
```

2.1.2.2 Accessing Help from Keypad Mode

From keypad mode, press the Help key to get a diagram of the keypad functions. The screen goes blank for a few seconds before the diagram appears.

After EDT displays the diagram, you can press any keypad key to see a description of what that key does. If you press keypad key number 5, for example, descriptions of the BACKUP and TOP keypad functions appear on the screen. At the end of each description are directions for seeing the whole keypad diagram again (by pressing Return), reading about other functions (by pressing another keypad key), or returning to your editing session.

For help on a control key sequence such as Ctrl/A, press both the control key and the keyboard key. For help on a Gold key sequence, such as Gold-A, press only the keyboard key, not the Gold key.

Note: The help diagrams do not display any keyboard or keypad keys that you have redefined.

2.1.2.3 Accessing Help from Nokeypad Mode

If you are in nokeypad mode and want to get help information about nokeypad commands, type EXIT, press Return, and enter HELP CHANGE at the asterisk prompt. After the help entry on CHANGE, there are five subtopics:

- ENTITIES
- HARDCOPY
- KEYPAD
- NOKEYPAD
- SUBCOMMANDS

The ENTITIES subtopic contains a list of the nokeypad entity subtopics: character, word, line, range, sentence, page, paragraph, select, vertical, and string. When you want information about one of these entities, include the entity name (for example, sentence) as the last HELP command subtopic:

```
* HELP CHANGE ENTITIES SENTENCE
```

The HARDCOPY, KEYPAD, and SCREEN subtopics provide brief descriptions of hardcopy, change mode, keypad mode, and screen editing. These do not have further subtopics.

Further information about nokeypad commands is contained in the SUBCOMMANDS subtopic. For a complete list of the subtopics covered, enter the following:

```
* HELP CHANGE SUBCOMMANDS
```

To leave help and return to your editing session, enter the CHANGE command at the asterisk, as follows:

```
* CHANGE
```

2.1.3 Recovering from System Interruptions

If your EDT editing session is interrupted, you can recover your work by using a journal file. Journal files and the method for recovering from interruptions are discussed in Section 2.7.

2.1.4 Moving from Mode to Mode

Once you become familiar with the three modes in EDT, you can switch from one mode to another to perform your editing tasks. The following table lists the commands you need to know to move from mode to mode:

From	To	Command
Line mode	Keypad mode	*CHANGE
Line mode	Nokeypad mode	*SETNOKEYPAD *CHANGE
Keypad mode	Line mode	Ctrl/Z
Keypad mode	Nokeypad mode	Ctrl/Z *SET NOKEYPAD *CHANGE
Nokeypad mode	Line mode	EX *
Nokeypad mode	Keypad mode	EX *SET KEYPAD *CHANGE

The following example demonstrates how to invoke EDT to create a new file named FUN.FUN, move to line mode (by default), and then move to keypad mode:

```
$ EDIT FUN.FUN ❶
Input file does not exist
[EOB]
* CHANGE ❷
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

The following example demonstrates how to invoke EDT to create a new file named WHY.NOT, enter line mode (by default), enter keypad mode, return to line mode, and, finally, enter nokeypad mode.

```
$ EDIT WHY.NOT ❶
Input file does not exist
[EOB]
* CHANGE ❷
[EOB]
Ctrl/Z ❸
* SET NOKEYPAD ❹
* CHANGE ❺
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

❸ Ctrl/Z returns you to line mode.

❹ The asterisk prompt indicates line mode and the SET NOKEYPAD command invokes nokeypad mode.

❺ The CHANGE command invokes nokeypad mode (only after you enter the SET NOKEYPAD command).

2.2 Using Keypad Mode

Keypad editing is available on VT300-series, VT200-series, VT100-series, and VT52 terminals. In keypad editing, the contents of a file are displayed on the screen as you edit. You can see the changes you make to a file as they take place.

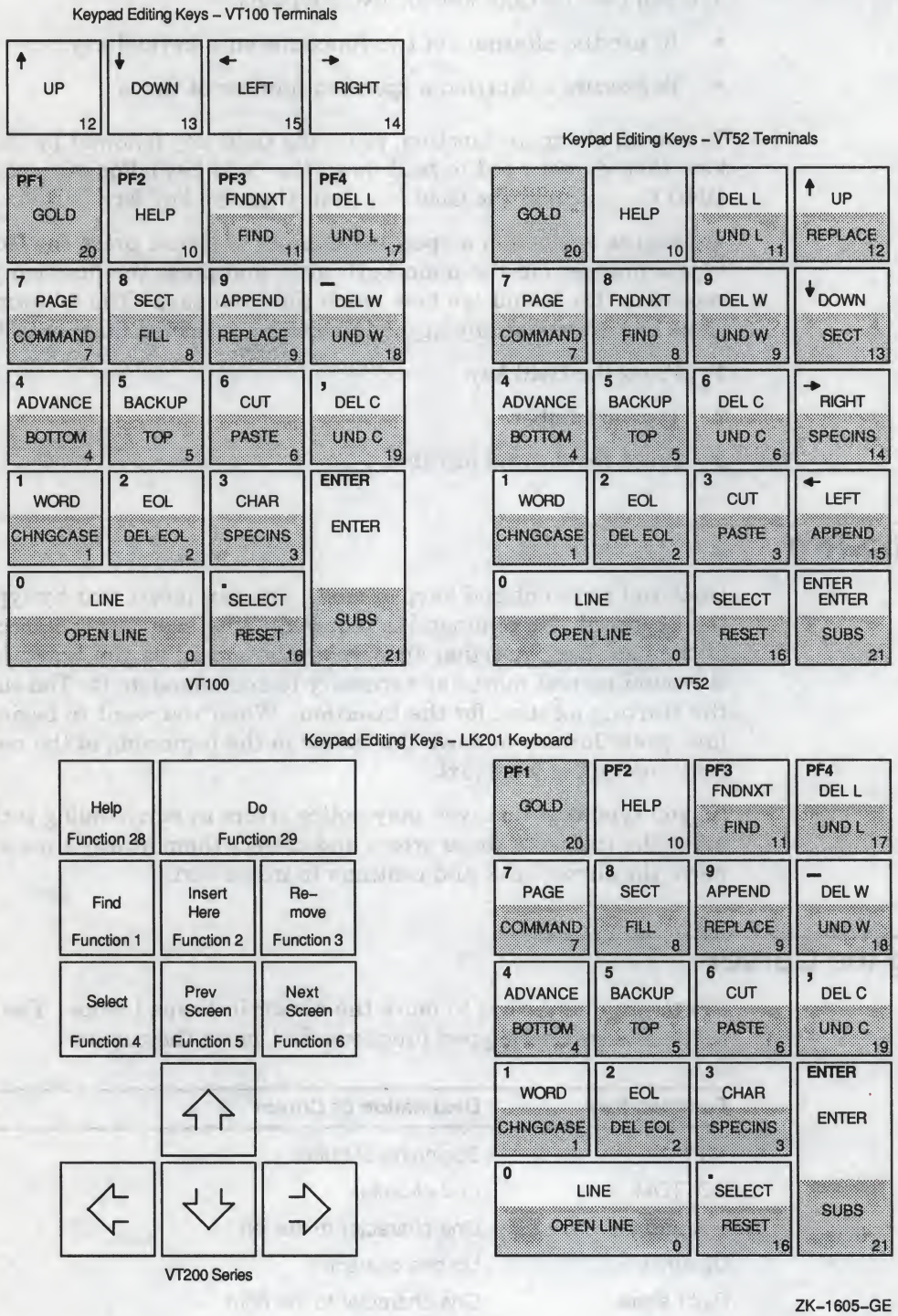
In keypad editing, you press keys to perform editing functions rather than type commands as is done in line and nokeypad editing.

2.2.1 Terminal Keypads

Figure 2–1 shows all the keypad functions available on VT52, VT100-series, and LK201 keyboards. (An LK201 keyboard is used with VT200-series and VT300-series terminals.) Each key in the keypad performs at least one editing command; many of the keys perform two. You can use the standard function (the upper half of the key) by pressing the key. You can use the alternate function (the lower, shaded half of the key) by pressing the Gold key before pressing the function key.

Notice that the LK201 keyboard includes a numeric keypad (like the keypad on the VT100-series terminal) and a six-key editing keypad with four arrow keys. (Two of the twenty available function keys, Help and Do, are also displayed.) See the *VAX EDT Reference Manual* for more detailed information about the LK201 function keys.

Figure 2–1 VT100, VT52, and LK201 Keypads



2.2.2 Using the Gold Key

You can use the Gold key for two purposes:

- To use the alternate of two functions on a keypad key
- To execute a function a specified number of times

To use an alternate function, press the Gold key followed by the desired key. (You do not need to hold down the Gold key.) For example, to use UND C, you press the Gold key first, then the key for UND C.

To execute a function a specified number of times, press the Gold key, type a number (on the main keyboard), and press the function you want repeated. Try it and see how much time you save. The following steps show how to repeat any keypad function a specified number of times:

1 Press the Gold key.

2 Type a number.

3 Press the desired function.

2.2.3 Inserting Text

Once you have entered keypad mode, you can insert text by typing it on the keyboard. No command is required. Whatever you type becomes part of the file. Your insertion appears on the screen as you type it, and the surrounding text moves as necessary to accommodate it. The cursor marks the starting location for the insertion. When you want to begin a new line, press Return to move the cursor to the beginning of the next line and continue typing your text.

As you type new text, you may notice errors in surrounding text. You can move the cursor to these errors and correct them at any time and then move the cursor back and continue to insert text.

2.2.4 Moving the Cursor

There are many ways to move the cursor in keypad mode. The following table lists several keypad functions that move the cursor:

Function Key	Destination of Cursor
TOP	Beginning-of-buffer
BOTTOM	End-of-buffer
Left arrow	One character to the left
Up arrow	Up one character
Right arrow	One character to the right
Down arrow	Down one character

The ADVANCE and BACKUP keypad functions cause the cursor to move in a forward or backward direction.

The following example demonstrates how to move the cursor to the beginning (or top) and to the end (or bottom) of the buffer, using the TOP and BOTTOM keypad functions.

Insert the following three lines of text:

```
Five golden rings,
four calling birds,
three French hens,
```

1. Press the Gold key followed by TOP to move the cursor to the *F* in the word *Five*.
2. Press the Gold key followed by BOTTOM. Notice the cursor moving to the left bracket of the end-of-buffer sign [EOB].
3. Press the Gold key followed again by TOP to move the cursor back to the *F*.

Repeat these steps to become familiar with the TOP and BOTTOM keypad functions.

The following example demonstrates how to move the cursor using the four arrow keys: up, down, left, right.

```
Under a toadstool crept a wee elf,
Out of the rain to shelter himself.
Under a toadstool all in a heap,
Sat a big dormouse, fast asleep.
```

1. Press the TOP keypad function (the Gold key followed by TOP) to move the cursor to the letter *U* in the word *Under*.
2. Press the right arrow 9 times, then press the down arrow once, moving the cursor to the *e* in the word *the*.
3. Press the left arrow 5 times, then press the down arrow once, moving the cursor to the *r* in the word *under*.
4. Press the right arrow 7 times, then press the down arrow once, moving the cursor to the first *o* in the word *dormouse*.
5. Press the left arrow 5 times, then press the up arrow 3 times, moving the cursor to the word *a*.

The next example demonstrates the keypad functions for TOP, LINE, EOL, WORD, and CHAR. Enter the following four lines of text:

```
ONE TWO THREE FOUR FIVE
SIX SEVEN EIGHT NINE
TEN ELEVEN TWELVE THIRTEEN
FOURTEEN
```

1. Press TOP to move the cursor to the first letter of the word *ONE*.
2. Press LINE once, moving the cursor to the *S* in the word *SIX*.
3. Press WORD twice, moving the cursor to the *E* in the word *EIGHT*. Then press EOL. The cursor moves to the end of the line.
4. Press WORD three times, moving the cursor to the *T* in the word *TWELVE*. Then press EOL. The cursor moves to the end of the line.

5 If you are using a VT100-series, a VT200-series, or a VT300-series terminal, press CHAR four times. The cursor moves to the *R* in the word *FOURTEEN*.

The following figure shows the approximate distances that some keypad commands move the cursor.

Keypad Function	Distance Keypad Function Moves Cursor
CHAR	#
WORD	######
EOL	###########
LINE	##
SECT	## ## ## ## ## ## ## ## ## ## ## ## ## ## ##

Keypad Function	Distance Keypad Function Moves Cursor
PAGE	## ##############################################

Note that the PREV SCREEN and NEXT SCREEN keys (on an LK201 keyboard) work like the SECT keypad function, moving the cursor back or ahead by 16 lines.

2.2.4.1 How EDT Views Words

You can determine the way that EDT recognizes a word, sentence, paragraph, or page. By default, EDT identifies a word as any group of contiguous characters starting at the current cursor position and continuing in the current direction until one of the following characters is encountered:

- Space ( )
- Horizontal tab (Tab)
- Line feed (LF)
- Vertical tab (VT)
- Form feed (FF)

- Carriage return (CR)
- Line terminator

For example, enter the following text and move the cursor throughout the paragraph by pressing the WORD key. Notice that the cursor stops on the first letter of each word.

```
This is the first line of text.
Is this the second?
This must be the third.
How about a fourth line?
And, a fifth, perhaps?
Will anyone take a sixth?
Number seven, please.
```

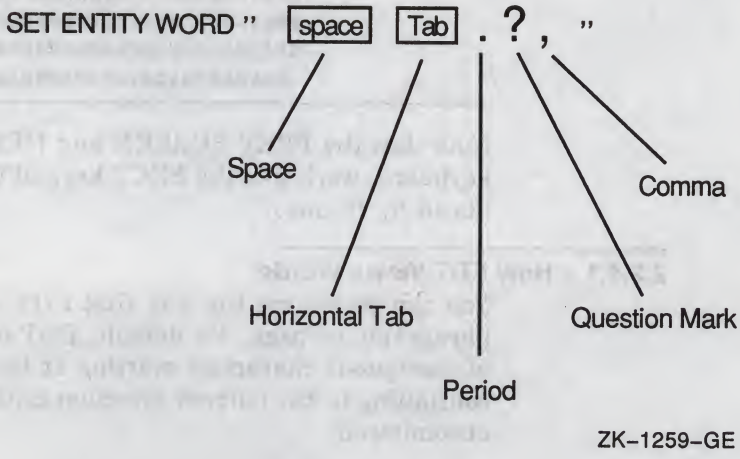
You can use the SET ENTITY WORD command to change the way EDT views words. If you want EDT to recognize periods, question marks, and commas, enter the following command line in your startup command file (for information about startup command files, see Section 2.11):

```
SET ENTITY WORD ".?,"
```

When you use the WORD key to move throughout the text again, notice that the cursor stops on all periods, question marks, and commas. For more information about the SET ENTITY command, see the *VAX EDT Reference Manual*.

Figure 2–2 shows the characters that can be used in the SET ENTITY command.

Figure 2–2 Using the SET ENTITY WORD Command



2.2.5 Deleting and Undeleting Text

You can delete text by character, word, and line. The DELETE and DEL C keys delete characters. DEL W and LINEFEED delete words or parts of words. DEL L, DEL EOL, and Ctrl/U delete single lines or parts of lines. The deleted text is stored in one of three buffers so that you can restore

it using the UND commands (UND C, UND W, and UND L). A buffer is a temporary holding area for text. The character buffer contains the last character deleted; the word buffer contains the last word deleted; and the line buffer contains the last line deleted. Only the most recent deletion can be restored; you can restore this unit of text numerous times to any location. See Section 2.6 for more information about buffers.

Figure 2–3 shows the three buffers that EDT fills when you use the delete and undelete keys.

Figure 2–3 Three EDT Buffers Used for Deleting and Undeleting Text

Use the following keypad functions to delete text:

Keypad Function	What It Does
DELETE	Deletes the preceding character.
DEL C	Deletes the current character.
DEL W	Deletes to the end of the word.
LINE FEED	Deletes to the beginning of the previous word.
DEL L	Deletes to the next line.
Ctrl/U	Deletes from the beginning of the line to the cursor.
DEL EOL	Deletes from the cursor to the end or beginning of the line, depending on the current cursor direction.
CUT	Deletes the selected string from the current file and stores it in the PASTE buffer. The selected string is all the text between the cursor location when you pressed the SELECT keypad function and the position to which you moved the cursor. (Note that the Remove key on the LK201 keypad works like the CUT keypad function.)

Enter the following three lines of text:

```
CHARACTER
WORD WORD WORD WORD WORD
LINE LINE LINE LINE LINE LINE LINE LINE
```

The following steps show how to delete and restore characters:

1 Move the cursor to the first *C* in the word *CHARACTER*. Press DEL C. The C disappears.

2 Press the Gold key followed by UND C. The C reappears.

3 Move the cursor to the *H*. Press DEL C to make it disappear.

4 Then press the Gold key followed by UND C. The H is restored.

5 Press the Gold key followed by UND C again, another H appears.

Notice that the DEL C keypad function deletes the character directly at the cursor and the <⌫ (Delete) key (on the main keyboard) deletes the character immediately to the left of the cursor. You can use the UND C keypad function to restore the most recently deleted character in either case. Press the <⌫ key and the DEL C keypad function to see the difference between the two.

Use the following steps to delete and restore words:

1 Move the cursor to the *W* in *WORD*. Press DEL W. The word *WORD* disappears.

2 Press the Gold key followed by UND W to restore the word.

3 Press the Gold key followed by UND W again, another *WORD* appears.

Notice that the DEL W keypad function deletes to the end of the current word, and the Line Feed key (on the main keyboard) deletes to the beginning of the preceding word. Press the Line Feed key to see the difference between the DEL W keypad function and the Line Feed key. Again, you can use the UND W keypad function to restore the deleted word in either case.

To delete and restore entire lines of text, do the following:

1 Move the cursor to the beginning of the line of LINES. Press the DEL L keypad function. The entire line disappears.

2 Press the Gold key followed by UND L to restore the line.

3 Press the Gold key and UND L several times to create multiple lines.

Press Ctrl/U to delete text from the cursor to the beginning of the line. Notice that the cursor moves to the beginning of the line. Press UND L to restore the line.

2.2.6 Locating Text

Use the FIND and FNDNXT keypad functions to locate strings of text. When you press the Gold key followed by FIND, the following prompt appears at the bottom of the screen:

```
Search for:
```

Type the string of text you are looking for and press one of the following keys:

- ADVANCE
- BACKUP

- Enter
- Do

If you press ADVANCE, EDT searches in a forward direction for the specified string. If you press BACKUP, EDT searches in a backward direction. If you press Enter or Do, EDT searches in the direction already set.

Note: The ADVANCE and BACKUP keys also set the direction for subsequent EDT commands.

Enter the following text:

```
Ella will not be able to attend the concert
tonight. She has a sore throat.  Perhaps you could give
the ticket to somebody in your music class.  Ella wants
to see the program when she is feeling better.
```

Press the Gold key followed by the FIND keypad function. When you are prompted for a search string, enter the word *ticket*.

```
Search for:  ticket
```

Now press BACKUP. The search starts at the end of the text (where the cursor is located) and continues towards the beginning of the text until it finds the word *ticket*.

Next search for the word *program*. Press the Gold key followed by FIND. When you are prompted for the search string, enter the word *program*.

```
Search for:  program
```

If you press BACKUP again, you see the response *String was not found* because the word *program* is located after the word *ticket*. Now press ADVANCE followed by FNDNXT, and EDT finds the string.

Remember the following three items when you are entering a search string:

- EDT ignores diacritical marks and the case of letters while making searches (unless you enter the SET SEARCH EXACT command).
- <⌧ is the only key you can use to edit an incorrectly typed search string.
- To cancel a search string, press Ctrl/U.

If you want to find the next occurrence of the string you are searching for, use the FNDNXT keypad function. EDT searches in the direction already set. If EDT cannot find the string, it displays the message *String was not found*. You can reverse the direction of the search by pressing either ADVANCE or BACKUP before pressing FNDNXT.

Enter the following text, then press Gold and Find, and enter the search string *little* when prompted:

```
One little, two little, three little chickadees,
four little, five little, six little chickadees,
seven little, eight little, nine little chickadees,
ten little chickadees feeding.

Search for:  little
```

Because the cursor is at the end of the text, you must press the BACKUP keypad function to set the search in a backward direction. Now press FNDNXT nine times. EDT finds each occurrence of the search string. When the cursor arrives at the first *little*, press the ADVANCE keypad function to reverse the direction of the search. Every time you press FNDNXT, the cursor moves forward to the next occurrence of the word *little*.

2.2.7 Moving Text

You can move text using three different groups of keypad functions:

- Return and OPENLINE—Pressing Return causes both text and cursor to move down the screen line by line. If you press OPENLINE, the text moves down the screen line by line, but the cursor remains in the same place.
- DEL L, UND L, DEL W, UND W, DEL C, and UND C—Deleting a unit of text (character, word, or line) from one location, moving the cursor to another location, and then undeleting the text is a way to move small portions of text. (See Section 2.2 for more information about deleting and undeleting text.)
- CUT and PASTE (or Remove and Insert Here)—Cutting (or removing) a unit of text and pasting (or inserting) it in another location is the method you use for moving large portions of text. This method is described in the rest of this section.

Use the following three keypad functions to move text:

- SELECT
- CUT (or Remove)
- PASTE (or Insert Here)

Press SELECT to mark one end of the string of text that you want to delete or move. Then move the cursor either backward or forward to the other end of the string, and press CUT. Before you press CUT, you can correct a mistake by pressing the Gold key followed by RESET to cancel the select range and start over. The selected text is highlighted in reverse video.

Press CUT to delete the selected string of text from the current file and store it in the PASTE buffer. (See Section 2.6 for more information about the PASTE buffer.) The selected text is all the text between the cursor location when you pressed SELECT and the position to which you have moved the cursor.

If you press CUT before you press SELECT, you see the message *No select range active*.

To insert the contents of the PASTE buffer to the left of the cursor position, press Gold followed by PASTE.

To summarize, use the following steps to move text:

1. Place the cursor at the beginning of the text you want to move.
2. Press SELECT to mark the location.
3. Move the cursor to the end of the select range.
4. Press CUT to delete the text from its current position.
5. Move the cursor to the character just beyond where you want the text inserted.
6. Press the Gold key followed by PASTE.

Note: If you want to restore the select string to its original location after you perform Step 4, press the Gold key followed by PASTE.

Enter the following text:

```
january february march april
may june july november
december august september october
```

In order to move the string *august september october* after *july*, move the cursor to the *a* in *august*. Press SELECT. Then move the cursor to the *r* in *october*. Press CUT. The selected string disappears into the PASTE buffer. Now move the cursor after the word *july* and press Gold followed by PASTE (Gold-PASTE). A copy of the selected string in the PASTE buffer appears on the screen between the words *july* and *november*.

If you press Gold-PASTE again, you get another copy of the contents of the PASTE buffer. Every time you press Gold-PASTE, you get another copy of the string *august september october*. You can always get as many copies of the PASTE buffer as you want by pressing Gold-PASTE. However, once you perform another SELECT and CUT operation specifying a different string, the contents of the PASTE buffer change.

If you want to add to the text in the PASTE buffer, retaining the text already there, use APPEND. APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer.

For example, if your PASTE buffer contains the text *Wolfgang Amadeus* and you want to add the text *Mozart*, follow these steps:

1. Press SELECT.
2. Type the word *Mozart*. (Precede the word with a space.)
3. Press APPEND.

Now the PASTE buffer contains the text:

```
Wolfgang Amadeus Mozart
```

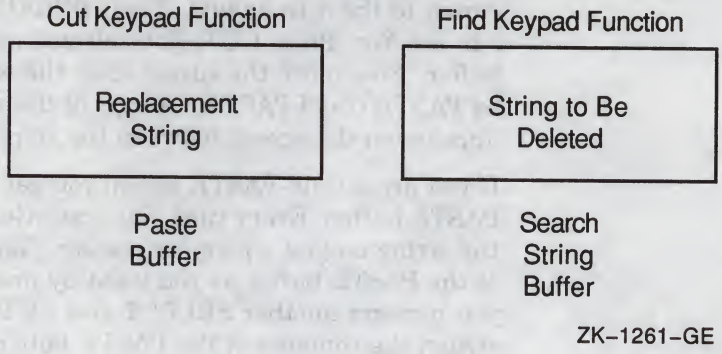
2.2.8 Substituting Text

You can use the SUBS keypad function or the REPLACE keypad function to replace one string of text with another.

When you make substitutions using SUBS, you are using both the PASTE buffer and the search string buffer. You use CUT to put the replacement string into the PASTE buffer and FIND to put the string you want to find and delete into the search string buffer. Figure 2–4 shows both buffers. The following five steps demonstrate how to use SUBS to make substitutions:

1. Press SELECT and enter the replacement text.
2. Press CUT. (The select range disappears into the PASTE buffer.)
3. Press the Gold key followed by FIND and enter the search string (the text you want to replace).
4. Press either ADVANCE or BACKUP to find the first occurrence of the search string in the direction you choose.
5. Press the Gold key followed by SUBS to exchange the existing text for the replacement text.

Figure 2–4 Two EDT Buffers Used for Substituting Text



When you use SUBS, EDT performs the substitution first and then moves to the next occurrence of the search string. If EDT cannot find another occurrence of the search string, it prints the message "String was not found." If you do not want to make a particular substitution, press the FNDNXT keypad function, and the cursor moves to the next occurrence of the search string. If you want to change that one, press SUBS again. Thus, you can move through the buffer pressing SUBS each time you want to make a replacement and pressing FNDNXT when you want EDT to leave the search string alone.

Note: You must use the FIND keypad function when you are making substitutions with SUBS because you are replacing text that matches the search string with the contents of the PASTE buffer. SUBS does not work correctly if you do not use FIND.

Enter the following text:

```
Susanne grabbed the red apple and gobbled it down.
Suddenly her face turned quite red.  Glancing towards
the red house she saw the huge tree bathed in red
leaves.
```

Now perform the following steps to substitute the word *green* for the word *red*:

1. Press SELECT and enter the word *green*.
2. Press CUT. (The word *green* disappears into the PASTE buffer.)
3. Press Gold followed by FIND and enter the word *red*.
4. Press either ADVANCE or BACKUP to find the first occurrence of *red* in the direction you choose.
5. Press SUBS to substitute the word *green* for the word *red*.
6. Press SUBS three more times.

Your resulting text should look like the following:

```
Susanne grabbed the green apple and gobbled it down.  Suddenly
her face turned quite green.  Glancing towards the green house
she saw the huge tree bathed in green leaves.
```

REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer. For example, if your PASTE buffer contains the words *paste paste paste*, and your select range contains the words *select select select*, press REPLACE to make your select range contain the words *paste paste paste*, as the following steps demonstrate:

1. Press SELECT.
2. Type the words *paste paste paste*.
3. Press CUT. Now your PASTE buffer contains the words *paste paste paste*.
4. Press SELECT again.
5. Type the words *select select select*.
6. Press REPLACE. Now your select range contains the words *paste paste paste*.

2.2.9 Five More Keys to Use with the Gold Key

Five keypad functions associated with the Gold key are summarized in the following table:

Keypad Function	What It Does
CHNGCASE	Reverses the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase.

Keypad Function	What It Does
COMMAND	Enables you to enter a line-mode command from keypad mode.
FILL	Reorganizes the select range so that the maximum number of whole words can fit within the current line width.
RESET	Changes the following conditions of your editing session: • Cancels an active select range. • Empties the search buffer so that there is no current search string. • Sets EDT's current direction to ADVANCE. • Sets EDT to the default DMOV state. (DMOV is a Nokeypad command that returns your editing session to EDT's default state, where EDT does not alter the case of letters during move operations.)
SPECINS (special insert)	Enables you to insert any character from the DEC Multinational Character Set into your text using the character's decimal equivalent value. For information about the DEC Multinational Character Set, see the *VAX EDT Reference Manual.*

For more information about all the available keypad keys, see the *VAX EDT Reference Manual.*

2.3 Using Line Mode

You can use EDT's line editing facility with any interactive terminal—hardcopy or screen. Line editing uses the line as its point of reference. EDT moves through the text line by line, not character by character as in the two other editing modes. Line editing commands are particularly useful for manipulating large blocks of text.

2.3.1 Line Numbers

To help you locate and edit text, EDT assigns line numbers. These line numbers are not part of the text and are not kept when you end an editing session. To see the line numbers of an already existing file, enter the TYPE WHOLE command after the asterisk prompt (*). Notice that the text you enter is indented 12 spaces.

The following example demonstrates how to display your line numbers:

```
*  TYPE WHOLE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    5           fivefivefivefive
[EOB]
*
```

Line numbers have the following characteristics:

- They are assigned to every line in every buffer in every editing session, including newly inserted lines and text added with the INCLUDE command.
- They start with 1 and increment by one, unless otherwise specified with the RESEQUENCE/SEQUENCE command.
- They are decimal numbers if newly inserted.
- They are removed by the EXIT command, unless otherwise specified.
- They can be renumbered in increments of one or more with the RESEQUENCE command.

When you insert new text, EDT numbers the lines using decimal numbers. For example, if you add a line of text between lines 13 and 14, it is numbered 13.1. To avoid any visual confusion when working with decimal numbers, enter the RESEQUENCE command after the asterisk prompt to change the sequence of line numbers. When you enter RESEQUENCE, EDT renumbers all the lines from the cursor to the end of the buffer in increments of one. You can specify a range of lines within the file by entering a range after typing RESEQUENCE.

If you insert text after the fourth line in the following example and enter the TYPE WHOLE command again, you will see the newly inserted text preceded by decimal numbers. To change the sequence of line numbers, enter the RESEQUENCE command.

```
* TYPE WHOLE
     1          oneoneoneoneoneoneone
     2          twotwotwotwotwo
     3          threethreethree
     4          fourfourfourfour
     4.1        four_point_one
     4.2        four_point_two
     4.3        four_point_three
     5          fivefivefivefive
[EOB]
* RESEQUENCE
     1          oneoneoneoneoneoneone
     2          twotwotwotwotwo
     3          threethreethree
     4          fourfourfourfour
     5          four_point_one
     6          four_point_two
     7          four_point_three
     8          fivefivefivefive
[EOB]
*
```

The /SEQUENCE qualifier determines the increments EDT uses to number lines. When you enter the following command and include two numbers separated by a colon (n:m), EDT numbers the lines in the buffer, assigning number *n* to the first line and incrementing by number *m*:

```
* RESEQUENCE/SEQUENCE:n:m
```

The following example resequences the contents of the buffer, assigning number 6 to the first line and incrementing by four:

```
* RESEQUENCE/SEQUENCE:6:4 Return
3 lines resequenced
* TYPE WHOLE Return
     6          Washington, Maine
    10          Vermont, New York
    14          Ohio, New Mexico
    18          Colorado, Virginia
    22          Florida, Arizona
    26          Alabama, Oregon
    30          Minnesota, New Hampshire
    34          Texas, California
    38          Delaware, Michigan
[EOB]
*
```

2.3.2 Inserting Text

When you want to insert text, enter the INSERT command after the asterisk prompt (*). The cursor indents 16 spaces and waits for you to start typing. (Indenting is on the screen or paper only and does not become part of the edited text.) You can enter as many lines as you want. Each time you come to the end of a line, press Return to move to the beginning of the next line.

As you enter a line, you can delete characters by pressing the <⌫ (Delete) key. You can delete a portion of a line, from the cursor to the left margin, by pressing Ctrl/U. However, when you end a line with Return, you can no longer delete characters from that line in insert mode. (See Section 2.3 for more information about deleting text.) When you finish entering text, press Ctrl/Z. EDT echoes ^Z on the screen and returns you to the asterisk prompt. To display the text you just inserted, enter the TYPE WHOLE command after the prompt.

Note: EDT, which inserts text in front of the current line, is different from many other text editors that insert text following the current line.

The following example demonstrates how to insert text:

```
$ EDIT NEWFILE.DAT
Input file does not exist
[EOB]
* INSERT
                This is the first line that I would like
                to type.  At the moment there is no file
                named newfile.dat in my directory.  When
                I finish my editing session and type the
                EXIT command, EDT will save a copy of
                this text in my directory under the name
                newfile.dat.
[EOB]
* EXIT
DBA0:[GLOVER]NEWFILE.DAT;1 5 lines
```

2.3.3 Ranges

When you want a line-mode command to affect a specific part of the buffer, you must enter a range. Ranges can specify one or more lines. In addition, the multiple-line ranges can be contiguous or noncontiguous.

The following table describes the different ranges you can specify when editing in line mode:

Range Type	Description
period (.)	Current line
number	EDT line number
'string'	Next line containing the quoted string
BEGIN	First line of the buffer
END	After the last line in the buffer ([EOB])
LAST	Last line EDT was at in the previous buffer
WHOLE	Entire buffer
BEFORE	All lines in the buffer before the current line
REST	All lines in the buffer starting with the current line and ending with the last line

The following examples demonstrate each range type just listed:

EXAMPLES

1
```
TYPE
```
This command displays the current line.

2
```
TYPE 35
```
This command displays line 35.

3
```
TYPE December
```
This command displays the first line it encounters containing *December*.

4
```
TYPE BEGIN
```
This command displays the first line of the current buffer.

5
```
TYPE END
```
This command displays the [EOB] mark.

6
```
TYPE LAST
```
This command displays the most recent line of text you displayed. The LAST specifier can be used only by itself; EDT does not accept LAST with other range specifiers or symbols or with buffer names.

7
```
TYPE WHOLE
```
This command displays every line in the current buffer.

8 TYPE BEFORE

This command displays the group of lines in the current buffer starting with the first line and ending with the line just before the current line.

9 TYPE REST

This command displays the group of lines in the current buffer starting with the current line and ending with the last line in the buffer.

You can use the following symbols and words with the range types listed above while editing in line mode:

Symbol/Word	Description
, or AND	Used to join noncontiguous ranges in a list; only single lines can be joined in this way
: or THRU	Indicates a group of lines starting with the first range specifier and ending with the second
n	Indicates the number of lines from the current line
# n or FOR n	Indicates the next "n" number of lines
+ "string" or "n"	Indicates that "string" or "n" refers to a line or lines after the current line
– "string" or "n"	Indicates that "string" or "n" refers to a line or lines before the current line
ALL "string" or "n"	Indicates that the command applies to all lines containing "string"

The following examples demonstrate the symbols and words just listed:

EXAMPLES

1 TYPE 3,6,8

This command displays lines 3, 6, and 8.

2 DELETE 4 AND 13

This command deletes lines 4 and 13.

3 TYPE 25:*Hartford*

This command displays the group of lines starting with line number 25 and ending with the line containing *Hartford*.

4 TYPE . THRU 150

This command displays all the lines in the group starting with the current line and stopping with line number 150.

5 DELETE 4#3

This command deletes line 4 and the three lines following line 4.

6 `TYPE . FOR 10`

This command displays the current line and the next ten lines; the number 10 refers to the tenth line after the current line.

7 `TYPE BEGIN +6`

This command displays the seventh line of the current buffer.

8 `DELETE -3 THRU`

This command deletes the current line and the three lines preceding it; the number –3 refers to the third line before the current line.

9 `TYPE "Dear"+2 THRU "Sincerely"-2`

This command displays the body of the letter, starting with the first paragraph and ending with the last.

10 `TYPE . THRU 1000 ALL "Date:"`

This command displays every line containing the string "Date:" that appears in the group of lines starting with the current line and ending with line number 1000.

2.3.4 Deleting Text

You can delete individual lines or groups of lines by using the DELETE command. After a delete operation, EDT displays the line following the last line deleted; this line is the new current line.

Use the following syntax:

```
* DELETE range
```

The following example demonstrates how to delete line 2 in a file named FUN.DAT:

```
$  EDIT FUN.DAT
*    1              This is the first line of a file named FUN.DAT.
* TYPE WHOLE
     1              This is the first line of a file named FUN.DAT.
     2              This is the second.
     3              This is the third,
     4              and the fourth.
* DELETE 2
1 line deleted
     3              This is the third,
* TYPE WHOLE
     1              This is the first line of a file named FUN.DAT.
     3              This is the third,
     4              and the fourth.
*
```

If you enter the DELETE command and do not specify a range, EDT deletes the current line.

When you delete a range of lines, EDT deletes the lines and displays a message stating the number of lines deleted. The message is followed by a display of the next line in the text buffer.

You can use the /QUERY qualifier if you want EDT to prompt you before deleting each line of a specified range. The prompt is a question mark. Answer the prompt with one of the following four responses:

- Y (Yes) Delete this line.
- N (No) Do not delete this line.
- A (All) Delete all remaining lines in the specified range.
- Q (Quit) Quit the delete operation.

You must enter one of these responses before EDT continues to the next line or exits from the delete operation. The following example demonstrates the /QUERY qualifier with the DELETE command:

```
* DELETE 7 THRU 13/QUERY Return
    7           This is the first line of the range.
? N Return
    8           This is the second line of the range.
? Y Return
1 line deleted
    9           This is the third line.
? A Return
5 lines deleted
    14          This line is not in the range.
*
```

2.3.5 Substituting Text

When you want to substitute one string for another, use either the SUBSTITUTE or SUBSTITUTE NEXT command. These are the only line editing commands that can alter text within a line, as opposed to changing the entire line. The SUBSTITUTE command operates on the current line or on a specified range. It has the capacity to make a global substitution; that is, you can replace every occurrence of one string with another by using only one line-mode command. The SUBSTITUTE NEXT command makes a substitution at the next occurrence of the string within the buffer.

The syntax for the SUBSTITUTE command is as follows:

```
* SUBSTITUTE /old-string/new-string/[range] [/QUERY]
```

To substitute the string GOOD for BAD throughout a buffer, enter the line-mode command SUBSTITUTE, followed by the old string and the new string. Separate all three with the same delimiter. (A delimiter is a character that marks the beginning or end of a string.) To apply the command to the entire buffer, specify WHOLE as the range parameter. When the operation completes, EDT supplies a message indicating how many substitutions were made.

Enter the following text:

```
He felt bad.  He had had a bad time in a bad part of
the city. How did he know that all the stores would
have bad merchandise. So, he had bad luck trying to
find the perfect gift.  His stomach felt bad after a
bad lunch.
```

Enter the following command:

```
* SUBSTITUTE/BAD/GOOD/WHOLE
```

The resulting text will look like this:

```
He felt GOOD.  He had had a GOOD time in a GOOD part of
the city. How did he know that all the stores would
have GOOD merchandise. So, he had GOOD luck trying to
find the perfect gift.  His stomach felt GOOD after a
GOOD lunch.
```

Slashes are not the only characters you can use to delimit strings. Most nonalphanumeric characters will work, as long as the delimiters are matched and do not occur in either string. For example, the following command substitutes the string A/3 for A/2 in the current line, using dollar signs ($) as delimiters:

```
* SUBSTITUTE$A/2$A/3$
    25          SIZE = A/3;
1 substitution
*
```

Note that a global substitution replaces all occurrences of the string, regardless of case, diacritical marks, or surrounding characters. If you want EDT to search for exact comparisons of case, enter the SET SEARCH command and specify the parameter EXACT. (For information about the SET SEARCH command, see the *VAX EDT Reference Manual*.)

If the search string occurs in the middle of a longer string, the substitution will still be made. For instance, a global substitution of IN for AT would change all words containing the string AT. (LATER would become LINER, THAT would become THIN, SAT would become SIN, and so on.) To get EDT to prompt you before each substitution, use the /QUERY qualifier with the SUBSTITUTE command:

```
* SUBSTITUTE/ in / at /WHOLE/QUERY
    1       He was in the point of no return.
? y
    1       He was at the point of no return.
1 substitution
```

EDT prompts you for one of the following responses before each occurrence of the search string:

Y—Yes, do the substitution
N—No, do not do the substitution
Q—Quit, terminate the command
A—All, do the rest of the substitutions without query

2.3.6 Moving Text from One Location to Another

You can use the MOVE and COPY commands to move one or more lines of text from one place to another. The effect of these commands is similar; the only difference is that the COPY command does not delete the text from its original location, whereas the MOVE command does.

The syntax for the MOVE command is as follows:

```
* MOVE first range TO second range/QUERY
```

When you use the MOVE command, EDT moves the lines in the first range above the line in the second range. EDT deletes the original copy of the text. For example, if you want to move lines 43 through 56 to above line 13, enter the following command:

```
* MOVE 43 THRU 56 TO 13 Return
```

In the following example, EDT moves the current line to above line 65:

```
* MOVE TO 65 Return
```

The second range always refers to a single line. EDT inserts the line or lines specified by the first range just above the line specified by the second range. The lines that are moved are renumbered to be consistent with their new location.

You can use the /QUERY qualifier with the MOVE and COPY commands to verify each line to be inserted in the range. (See Section 2.3 for more information about the /QUERY qualifier.)

Use the following COPY command to copy text without deleting the original text:

```
* COPY first-range TO second-range/QUERY/DUPLICATE:n
```

When you use the COPY command, EDT copies the lines in the first range to a location above the line in the second range. For example, if you want to copy lines 16 through 24 to above line 3, enter the following command:

```
* COPY 16 THRU 24 TO 3
```

In the following example, EDT copies lines 15 through 60 to the position just before line 95.

```
* COPY 15 THRU 60 TO 95
```

You can use the /DUPLICATE qualifier with the COPY command if you want to insert the range of text more than once. In the following example, EDT copies the first line and places it just above line 65. This operation is repeated nine times.

```
* COPY BEGIN TO 65/DUPLICATE:9 Return
1 line copied 9 times
*
```

2.3.7 Replacing Text

The REPLACE command combines the DELETE and INSERT functions in one command. You can use REPLACE when you need to delete a block of text and want to type new text in that same location. The syntax of the REPLACE command is as follows:

```
* REPLACE range Return text Ctrl/Z
```

When you use the REPLACE command, EDT deletes the line or lines specified by the range. As with the DELETE command, EDT prints a message telling you how many lines you have removed.

If you supply no specifiers with the command, EDT deletes the current line. As soon as EDT finishes deleting the specified line or lines, it shifts to the insert state. (The cursor moves to the right, just as it does when you enter the INSERT command.)

Anything you type after pressing Return is inserted. When you finish typing the new text, press Ctrl/Z to signal EDT that you want to return to the asterisk prompt (*).

The following REPLACE command deletes the current line and shifts to the insert state:

```
* TYPE
   17      This is the current line.
* REPLACE
1 line deleted
            I have just deleted the current line and
            am adding new lines to my text.  Ctrl/Z

   18      This is the next line.
* TYPE 16 THRU 18
   16      This is the line before the current line.
   16.1    I have just deleted the current line and
   16.2    am adding new lines to my text.
   18      This is the next line.
```

2.4 Using Nokeypad Mode

You can use nokeypad editing on VT300-series, VT200-series, VT100-series, and VT52 terminals.

The commands you use in nokeypad editing are English words or abbreviations that you enter from the main keyboard to move the cursor and edit text. As you type nokeypad commands, they are displayed on the lower left portion of the screen. As in line mode, you press Return to process the commands. You can join several commands on a single line and process all of them by pressing Return. Note the following characteristics of the cursor in nokeypad mode:

- The cursor is at the top of the screen when you enter nokeypad mode.
- You can move the cursor by using the arrow keys.
- When you do not specify a range with your nokeypad command, you can affect the text at the cursor.

When you type two commands on the same line, such as 3W D-C, you have the option of separating them with a space (for the sake of clarity), even though EDT does not require one.

Nokeypad commands can be used to define keys for keypad mode editing.

See the *VAX EDT Reference Manual* for information about using nokeypad mode.

2.5 Modifying an EDT Environment

EDT provides many tools for modifying your own EDT environment. You can change the appearance of your screen display, the behavior of lines of text as you insert them, and the mode in which you are working. Section 2.5.1 discusses some of the SET commands provided by EDT. See the *VAX EDT Reference Manual* for descriptions of all the available SET commands.

2.5.1 Using SET Commands

You can use the SET commands to change the way EDT works. Some SET commands affect the display of text on your screen. For example, you can specify the number of lines you want displayed (SET LINES number) and whether or not the lines have line numbers (SET [NO]NUMBERS). You can also specify that EDT make an exact search (SET SEARCH EXACT).

By default, your screen contains 22 lines. If you want to decrease the screen display to 5 lines, enter the command SET LINES 5. If you are editing at slow data rates, you can increase your editing speed by decreasing the number of lines displayed on your screen.

When you are working in line mode, EDT displays line numbers by default. If you do not want the numbers to appear on the screen or paper, enter the command SET NONUMBERS.

By default, when you press the Gold key followed by the FIND key, EDT searches for the specified string, disregarding the case of letters. For example, if you enter the search string *course,* EDT finds every occurrence of the word (for example, *course, Course, COURSE*). But, when you enter SET SEARCH EXACT, EDT only finds occurrences of the word that exactly match the specified string (*course*).

You can use the following line-mode commands to enter one of the other modes:

```
SET KEYPAD
SET NOKEYPAD
```

In a startup command file, you would use the following SET commands to determine your own default editing mode:

```
SET MODE LINE
SET MODE CHANGE
```

For example, to enter nokeypad mode from keypad mode, enter the following command after pressing Gold-COMMAND:

```
Command:  SET NOKEYPAD
```

One SET command, SET QUIET, even controls the sound of your terminal. You can use this command to suppress the terminal bell that signals an error.

2.5.2 Using SHOW Commands to See What Is Set

EDT provides the SHOW commands to enable you to see what is set and what is not set. For every SET command there is a SHOW command. If you want to see the number of lines on your screen, enter the SHOW LINES command. If you want to see whether EDT is performing an exact search, enter the SHOW SEARCH command.

For example, if you want to check the number of lines displayed on your screen, enter the following command:

```
* SHOW LINES
22
```

Or, if you want to see whether line numbers will be displayed in line mode, enter the following command:

```
* SHOW NUMBERS
numbers
```

The following table lists the SET commands discussed so far along with their corresponding SHOW commands:

SET Command	SHOW Command
SET KEYPAD SET NOKEYPAD	SHOW KEYPAD
SET LINES	SHOW LINES
SET MODE LINE SET MODE CHANGE	SHOW MODE
SET NUMBERS SET NONUMBERS	SHOW NUMBERS
SET QUIET SET NOQUIET	SHOW QUIET
SET SCREEN	SHOW SCREEN
SET TRUNCATE SET NOTRUNCATE	SHOW TRUNCATE
SET WRAP SET NOWRAP	SHOW WRAP

For a complete list of the SHOW commands provided by EDT, see the *VAX EDT Reference Manual*.

2.6 Using Buffers

Buffers are temporary holding areas for text. They enable you to work with more than one file at a time. You can use the buffers available in EDT to do the following:

- Divide one or more files into sections
- Move part or all of another file into your editing session
- Create a file from part or all of the text in a buffer

When you start an editing session, EDT automatically provides a buffer called MAIN. The MAIN buffer serves as the work area for text you insert and edit. If you are editing a file that already exists, EDT puts a copy of the contents of the file into this MAIN buffer.

The MAIN buffer has the following characteristics:

- MAIN is provided automatically when you invoke EDT.
- MAIN exists during your entire editing session.
- MAIN cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

The other buffer that EDT provides automatically is called the PASTE buffer. When you use the CUT keypad function, the deleted text goes into the PASTE buffer. Every time you perform a new CUT operation, EDT clears the PASTE buffer and replaces its contents with the newly deleted text.

The PASTE buffer has the following characteristics:

- The PASTE buffer is provided automatically when you begin your editing session.
- The PASTE buffer can be edited.
- The PASTE buffer can be loaded with the contents of another file. (Use the INCLUDE command.)
- The PASTE buffer cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

2.6.1 Viewing Existing Buffers

If you want to see a list of the buffers in your editing session, enter the line-mode command SHOW BUFFER. If you are working in your MAIN buffer and you have not used the CUT command or created any new buffers, you get the following display:

```
=MAIN    23      lines
 PASTE   no      lines
```

(The number of lines, 23, used in this example is arbitrary.)

These two buffers, MAIN and PASTE, are always displayed when you enter the SHOW BUFFER command because they always exist and you cannot delete them.

If you have just used the CUT and PASTE commands to move three lines of text and you enter the SHOW BUFFER command again, the display indicates those three lines in the PASTE buffer:

```
=MAIN    20      lines
 PASTE   3       lines
```

The current buffer (the buffer in which you are working) is preceded by an equal sign. In the example above, MAIN is the current buffer. If you cannot remember which buffer you are working in, enter the SHOW BUFFER command.

Sometimes you will notice an asterisk (*) following the line count for the MAIN buffer. This asterisk indicates that EDT has not had a chance to read through the entire input file and has seen only as many lines as are indicated.

2.6.2 Creating Buffers

To create a buffer from keypad mode, press the Gold key, followed by the COMMAND function. When EDT prompts you with *Command:*, enter the FIND command, then an equal sign followed immediately by the buffer name of your choice. A buffer name can contain any alphanumeric characters, but it must begin with a letter. The only punctuation character you can use in a buffer name is an underscore.

To create a buffer named BIRD, enter the following line after the command prompt:

```
Command:  FIND=BIRD
```

When you enter this command, the screen clears except for the [EOB] symbol, indicating that the current buffer, BIRD, is empty. After you press Return, you can insert and edit text just as you would in the MAIN buffer.

To create a buffer from line mode, enter the FIND command, an equal sign, and the buffer name after the asterisk prompt. For example:

```
* FIND=BIRD
```

To return to the MAIN buffer, type FIND=MAIN. The buffer named BIRD remains intact until you exit from the file, when only the MAIN buffer is saved.

You can also create a buffer during a MOVE operation. For example, to move lines 1 through 17 to a new buffer named AINGE, enter the following line-mode command:

```
* MOVE 1:17 TO =AINGE
```

2.6.3 Deleting Buffers

Use the line-mode command CLEAR to delete buffers during an editing session. For example, to delete a buffer named WALTON, type the following command:

```
* CLEAR WALTON
```

You cannot delete the buffers MAIN and PASTE, only their contents. If you enter the command CLEAR MAIN, the contents of your MAIN buffer disappear, but the buffer name remains. When you exit from EDT, the contents of all the existing buffers, except MAIN, are deleted.

2.6.4 Copying Text from One Buffer to Another Buffer

To copy text from one buffer to another, enter the COPY command. For example, to copy the contents of a buffer named BIRD to a buffer named PARISH, enter the following command:

```
* COPY =BIRD TO =PARISH
```

When you complete this operation, the current buffer will be PARISH.

2.6.5 Copying Text from a File into a Buffer

To copy text from a file outside of EDT into a buffer, use the INCLUDE command. For example, to copy the contents of a file named JOHNSON.DAT to your MAIN buffer, type the following command:

```
* INCLUDE JOHNSON.DAT =MAIN
```

2.6.6 Copying Text from a Buffer to a File

To copy text from a buffer to a file, use the WRITE command. You can use this command to copy text to a new file without affecting your editing session. For example, the following command puts a copy of lines 23 through the end of the current buffer into a file that you name MCHALE.DAT:

```
* WRITE MCHALE.DAT 23:END
```

The following table lists the three commands you need to copy text between buffers and external files:

Direction	Command
One buffer to another	* COPY =other_buffer TO =buffer
External file to buffer	* INCLUDE alien_file.typ =buffer
Buffer to external file	* WRITE alien_file.typ n:n

2.7 Recovering a Lost Editing Session

While you are editing or inserting text, EDT is keeping track of every keystroke you enter at your terminal. EDT records this information in a file called a journal file. Unless you specify otherwise, this journal file is deleted as soon as you enter the EXIT or QUIT command. However, when you experience a system interruption, the journal file is not deleted.

The journal file does not contain a version of your text. Rather, it contains a record of the keystrokes you entered during the session. By combining the journal file with the text that you had at the beginning of your session, you can recover your session to a point just before the interruption.

Note: Sometimes, the last few keystrokes are missing. This is normal. No work from earlier in your session will be omitted.

The following command line demonstrates how to recover a file named TRAVEL.SDML after the editing session has been interrupted:

```
$ EDIT/RECOVER TRAVEL.SDML
```

When you work with journal files, you will notice that they have a file type of JOU. The file name is the same as the file you were editing. The journal file for TRAVEL.SDML is TRAVEL.JOU. And, the journal file for ASCONA.LIS is ASCONA.JOU.

To recover an editing session you had begun with the command EDIT JONES.DAT, enter the following command line:

```
$ EDIT/RECOVER JONES.DAT
```

Notice that you enter the name of the file you were editing, not the name of the journal file. If you enter the JOU file type, you edit the journal file. After you enter the EDIT/RECOVER command, EDT replays the editing session. When it finishes, you can continue editing.

You can start an editing session and interrupt it to see how the /RECOVER qualifier works with the following steps:

1. Invoke EDT to create a file named NBA.FUN.
2. Insert text into NBA.FUN and perform a number of edits.
3. Press Ctrl/Y to end the editing session abruptly. (You will be at the DCL command level at this point.)
4. Enter the DCL command DIRECTORY to see the newly created journal file (NBA.JOU).
5. Invoke EDT again, using the /RECOVER qualifier to recover your lost session, by entering the command line:

   ```
   $ EDIT/RECOVER NBA.FUN
   ```
6. Be sure that your last few keystrokes were recovered and continue to edit the file. Then enter the EXIT or QUIT command to terminate your editing session.
7. Enter the DCL command DIRECTORY again. The journal file, NBA.JOU, is not there because you have exited normally from EDT.

2.8 Creating Columns and Layered Text

You can use EDT's tabbing facility in keypad mode to format text in two ways: in columns of eight and in layers.

2.8.1 Creating Columns of Eight

EDT has automatic tabs set for columns eight characters wide. Therefore, when you press Tab, your text moves to the nearest preset tab stop: column 9, 17, 25, 33, 41, 49, and so on. Tab always moves your text to the right regardless of EDT's current direction. If you want to move text back toward the left margin, use the <⌫ (Delete) key.

The following example demonstrates how to use Tab. Enter the following lines of text:

```
Tab each word in this sentence?
Tab each word in this sentence?
Tab each word in this sentence?
```

Move the cursor to the first letter of the first word and press Tab. Then move the cursor to the first letter of the second word and press Tab; then, to the third word and press Tab, and so on. When you are done, the sentences look as follows:

```
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
```

The words are organized in columns of eight characters. As long as you want to arrange text in columns of eight, you can use Tab.

2.8.2 Creating Layers of Text

If you have layered text, such as an outline, you can use the line-mode command SET TAB along with the Tab key to format the lines. When you enter the SET TAB command and specify a value (for example, 15), you can tab your first item over 15 spaces by pressing Tab. The following example shows a list of colors:

```
red
green
beige
blue
white
```

After you enter the following line-mode command, move the cursor to the first letter of each word in your list of colors and press Tab:

```
* SET TAB 15
```

Your tabbed list is shown below:

```
               red
               green
               beige
               blue
               white
```

The two most important things to remember when you are using SET TAB are the following:

- The SET TAB value affects only the first indentation on a line.
- The cursor must be at the left edge of the screen on the line being moved.

To format an outline, use the line-mode command SET TAB along with the keys Ctrl/E, Ctrl/D, and Tab to change the indentation level of your lines.

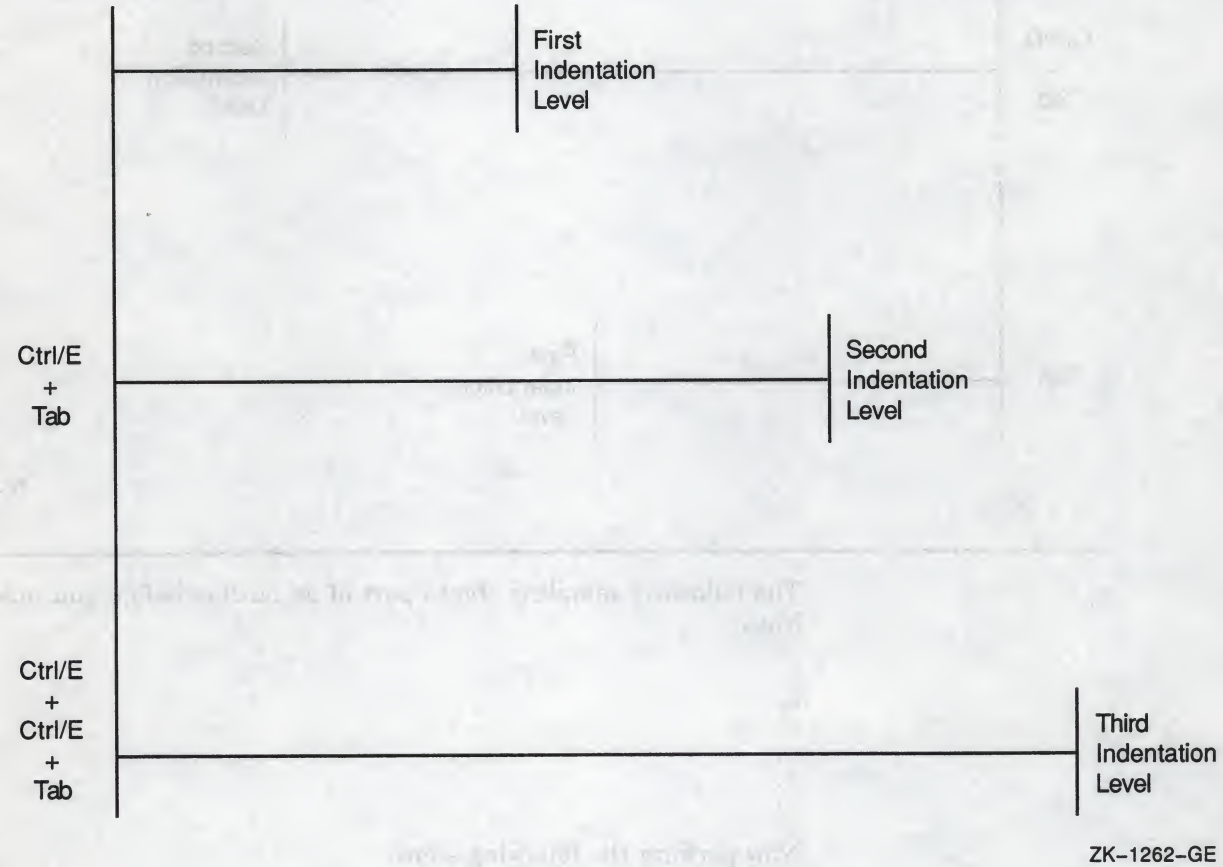
Enter the SET TAB command to set the indentation level. After you enter the SET TAB command, you can move the first line to be indented to the first indentation level by pressing Tab. To move a line of text to the second indentation level, press Ctrl/E. Then press Tab to actually move the line of text. You increase the indentation level of the text by pressing Ctrl/E, and you move the text to the desired level by pressing Tab.

For example, enter the command SET TAB 10, press Tab, and type the word TEN. Press Return. Then press Ctrl/E, followed by Tab, and type the word TWENTY. Press Return. Next, press Ctrl/E again, Tab, and type the word THIRTY. Press Return. Finally, press Ctrl/E, Tab, and type the word FORTY. Your display will look as follows:

```
TEN
        TWENTY
                THIRTY
                        FORTY
```

Figure 2–5 shows the keys Tab and Ctrl/E with their corresponding indentation levels.

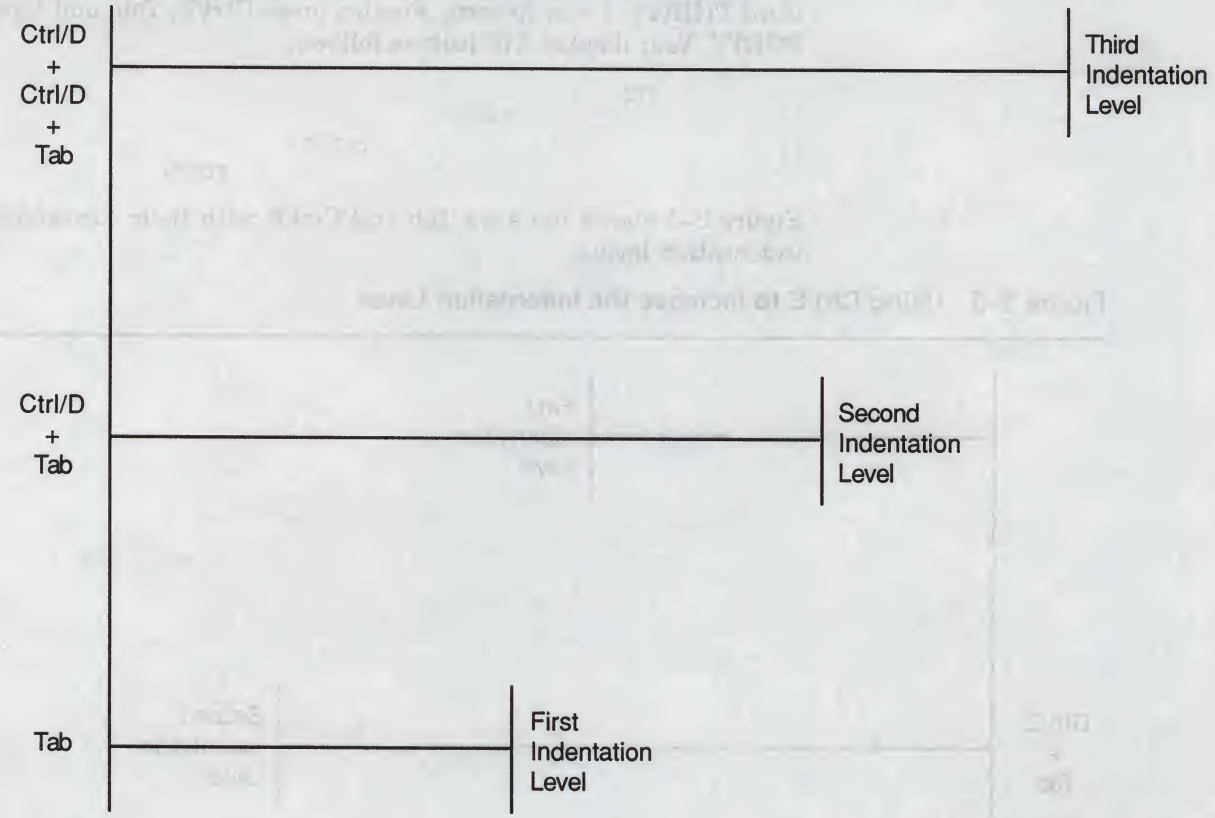
Figure 2–5 Using Ctrl/E to Increase the Indentation Level



Use Ctrl/D to select the next lower indentation level. Ctrl/D does the opposite of Ctrl/E. It lowers the indentation level count by the value of SET TAB. It does not move text back toward the left margin; rather, it reduces the number of spaces that text is moved to the right.

Figure 2–6 shows the keys Tab and Ctrl/D with their corresponding indentation levels.

Figure 2–6 Using Ctrl/D to Decrease the Indentation Level



The following sampling shows part of an outline before you indent the lines:

```
I.
A.
B.
1.
C.
II.
```

Now perform the following steps:

1. Choose the number of spaces you want to indent (for example, 5).
2. Enter the SET TAB command followed by that number.
3. Move the cursor to the Roman numeral I and press Tab.

4 Move the cursor to the letter A, press Ctrl/E, and press Tab.

5 Move the cursor to the letter B and press Tab.

6 Move the cursor to the number 1, press Ctrl/E, and press Tab.

7 Move the cursor to the letter C, press Ctrl/D, and press Tab.

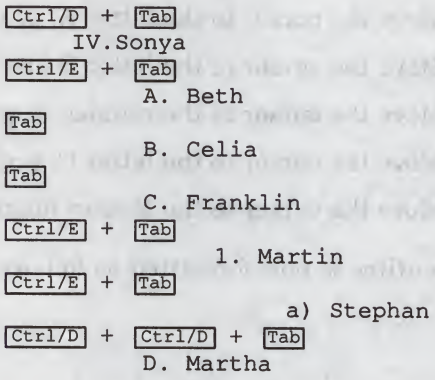
8 Move the cursor to the Roman numeral II, press Ctrl/D, and press Tab.

The outline is now formatted as follows:

```
I.
        A.
        B.
                1.
        C.
II.
```

The following shows a longer outline and the tabbing keys you press to create it:

```
                    Malcolm's Family Tree

Tab
    I. Malcolm
Ctrl/E + Tab
            A. Ian
Tab
            B. Tatiana
Tab
            C. Lars
Ctrl/E + Tab
                    1.  Stephanie
Tab
                    2.  Brian

Ctrl/D + Ctrl/D + Tab
    II.Frederick
Ctrl/E + Tab
            A. Louis
Ctrl/E + Tab
                    1.  Elsa

Ctrl/D + Ctrl/D + Tab
    III.Louisa
Ctrl/E + Tab
            A. Jeffrey
Tab
            B. Andrew
```



Note: Only two keypad tabbing functions actually move text: Tab and Ctrl/T. The remaining three functions (Ctrl/A, Ctrl/D, and Ctrl/E) determine the effect that the first two have on your text. Ctrl/A and Ctrl/T are discussed in the following sections.

2.8.3 Using Ctrl/A to Indent Text

You can use Ctrl/A to move a line of text to more than one indentation level without having to enter repeated tab increments (Ctrl/E). There are three steps to the process:

1. Enter a SET TAB command, specifying a value.
2. Move the cursor to one of the indentation levels. The indentation level must be a character position that is a multiple of the value you specified with SET TAB. For example, you could move the cursor to 20 or 25 if the tab size is 5.
3. Press Ctrl/A.

Once you choose the indentation level you want, and establish the position with Ctrl/A, you can use Tab to indent the line of text to that place.

If you want to indent text to a different indentation level after establishing a position with Ctrl/A, you can either use Ctrl/A again to reset the position or use Ctrl/E and Ctrl/D to adjust the indentation level up or down.

Note: If the cursor position is not evenly divisible by the SET TAB value when you press Ctrl/A, EDT displays the message *Could not align tabs with cursor.*

2.8.4 Using Ctrl/T to Indent Groups of Lines of Text

Use Ctrl/T to indent groups of lines using the current SET TAB value. The Ctrl/T function is similar to Ctrl/A, Ctrl/D, and Ctrl/E in the following ways:

- Ctrl/T has no effect unless you have specified a value with the SET TAB command.
- Ctrl/T moves entire lines of text.

On the other hand, Ctrl/T differs from these other Ctrl tabbing functions in the following ways:

- When you press Ctrl/T, you do not press Tab to make the block of text move to the right. The text moves automatically.
- The block of text is indented only the amount of the value established with the SET TAB command, without regard to the current indentation level.

To allow Ctrl/T to work correctly in EDT, you must disable DCL control over Ctrl/T. By default, DCL displays process statistics when you press Ctrl/T. To disable DCL control, enter the following command at the DCL command level:

```
$ SET NOCONTROL=T
```

To indent text using Ctrl/T, use the SELECT command to choose a range of lines, and then press Ctrl/T. EDT indents the range of lines only to the current SET TAB value. To indent text by multiples of the SET TAB value, use the Gold repeat function described in Section 2.2.2.

Note: All the functions, except for Tab, require that a SET TAB value be established for your editing session. EDT's default is SET NOTAB. As long as NOTAB is in effect, pressing Ctrl/A, Ctrl/D, Ctrl/E, or Ctrl/T has no effect on the text you are editing.

2.8.5 Looking at the Indentation Level

Use the SHOW TAB command to see the value specified with the SET TAB command and the current indentation level. For example, if you set your tab size to 15 and press Ctrl/E twice, EDT displays the following information when you enter the SHOW TAB command:

```
* SHOW TAB
tab size 15; tab level 3
```

For more information about the SHOW TAB command, see the *VAX EDT Reference Manual*.

2.9 Defining Keys

When you define or redefine a key, you change the function of that key. You can use combinations of nokeypad commands or existing keypad functions to dictate what a key does.

Some advantages of defining keys are as follows:

- You can take advantage of different entities available in nokeypad mode. For example, you can define keys to delete sentences, paragraphs, and pages.
- You can combine several nokeypad commands into one key definition. For example, you can define a key to find a word, delete it, and substitute several words in its place.

- You can access nokeypad commands that do not exist in keypad editing, such as SHL, SHR, CHGL, CHGU, and DATE.

2.9.1 Definition Procedure

EDT provides two commands to define keys: Ctrl/K and DEFINE KEY. The line-mode command DEFINE KEY allows you to put key definitions in startup command files and EDT macros, as well as to create key definitions during your EDT session. You can use Ctrl/K only when working in keypad mode.

2.9.1.1 Using Ctrl/K to Define a Key

To define a key in keypad mode, do the following:

1. Press Ctrl/K.
2. Press the key (or key sequence) you want to define (for example, Gold-A or Ctrl/H).
3. Type in the key definition. (Either enter a string of nokeypad commands or press a sequence of keypad function keys, or both.)
4. Type a period.
5. Press the Enter key.

To define keys, you need to be familiar with nokeypad commands. See the *VAX EDT Reference Manual* for more information.

The following example uses D and SEN. D is the delete command and SEN refers to a string of characters (a sentence) enclosed by delimiters. This example defines Ctrl/A to delete the sentence at your cursor.

```
Ctrl/K
Press the key you wish to define
Ctrl/A
Now enter the definition terminated by Enter
DSEN. Enter
```

The following example demonstrates how to define a key in keypad mode to draw a vertical line:

```
Ctrl/K
Press the key you wish to define
Ctrl/V
Now enter the definition terminated by Enter
22(+VI|^Z). Enter
```

In this example, Ctrl/K tells EDT to define the key combination Ctrl/V. The number 22 tells EDT to execute the commands in the parentheses 22 times. The definition for the down arrow is +V. The command I | tells EDT to put the vertical bar character into your text. The insert command is terminated by ^Z. The period ensures that the command will take effect as soon as you press the key.

Note: The ^Z is typed as a circumflex followed by the letter Z.

2.9.1.2 Using the DEFINE KEY Command

The DEFINE KEY command allows you to assign a different command to a key or assign a string of text to a key. To assign a command to a key, use the following syntax with the DEFINE KEY command:

```
* DEFINE KEY key name AS "command(s)."
```

Key name is the name of the key or its keypad number, for example, DEFINE KEY Gold F or DEFINE KEY 3. (See Figure 2–1 for keypad diagrams showing key names and keypad numbers. Note that you use the small number in the lower right-hand corner on the keypad diagram. For example, you would use the number 11 to redefine keypad key PF3.) Type the *command(s)*, which is the actual definition, completely in nokeypad syntax. Follow the nokeypad commands with a period and enclose the definition (and the period) in quotation marks or apostrophes.

Note: To use a key you have defined with the DEFINE KEY command, you must be in keypad mode. To enter keypad mode, see Section 2.1.4.

The steps for defining a key using the DEFINE KEY command are outlined below:

1. Enter line mode. (See Section 2.1.4 if you need help.)
2. Enter the DEFINE KEY command.
3. Type the key name you want to define followed by the word *AS* (for example, DEFINE KEY CONTROL B AS . . .).
4. Enter a string of nokeypad commands followed by a period. Enclose the string and the period in quotation marks (for example, DEFINE KEY CONTROL B AS *"D+NL."*).
5. Press Return.
6. Exit line mode. (See Section 2.1.4 if you need help.)

To assign a string of text to a key, use the following syntax with the DEFINE KEY command:

```
* DEFINE KEY key-name AS "IInsert your text here. ^Z."
```

In this example, the command to insert text is enclosed in quotation marks. The following components are in the quotation marks:

- The INSERT command (I).
- Text to be inserted.
- ^Z (to complete the insertion). Remember to enter the ^Z by typing the circumflex followed by the letter Z.
- A period to enable the command to take effect as soon as you press the key.

When you use the DEFINE KEY command to define Ctrl keys, for example Ctrl/A, type the word CONTROL:

```
* DEFINE KEY CONTROL A AS "command."
```

When you want to define the Gold key, for example Gold 5, type the word Gold:

```
* DEFINE KEY Gold 5 AS "command."
```

When you want to redefine a function key on the LK201 keyboard, type the word FUNCTION, as in the following example:

```
* DEFINE KEY FUNCTION 29 AS "command."
```

To create key definitions, you can consolidate several keypad functions into one key definition. The following table lists the key names you use when you define keys:

Key Name	VT100	VT52	LK201
20	PF1	red key	PF1
10	PF2	blue key	PF2
11	PF3	black key	PF3
17	PF4	-	PF4
0-9	keypad 0-9	keypad 0-9	keypad 0-9
16	period key	period key	period key
19	comma key	comma key	comma key
18	minus key	minus key	minus key
21	ENTER key	ENTER key	Enter key
FUNCTION 1	-	-	Find
FUNCTION 2	-	-	Insert Here
FUNCTION 3	-	-	Remove
FUNCTION 4	-	-	Select
FUNCTION 5	-	-	Previous Screen
FUNCTION 6	-	-	Next Screen
FUNCTION 28	-	-	Help
FUNCTION 29	-	-	Do

You can combine keypad functions to correct typing errors. For example, to correct reversed letters, you delete a character, press the right arrow key, and undelete the character. Suppose you type the word *social* as *oscial.* You would move the cursor to the *o,* delete it, press the right arrow key to move the cursor to the *c,* and undelete the *o.* The following table displays these steps translated into nokeypad commands:

Function	Nokeypad Commands
Delete a character	D+C
Move cursor one character to the right	+C
Undelete a character	UNDC

To define Ctrl/R to do all three commands at once (thus saving you keystrokes), enter the following line:

```
* DEFINE KEY CONTROL R AS "D+C +C UNDC."
```

Now, when you press Ctrl/R, EDT will reverse the two characters at the cursor.

2.9.2 Which Keys Can Be Defined

You can define all keypad keys, all function keys, and Gold with either a keypad key or a function key.

```
* DEFINE KEY 8 AS "command."
* DEFINE KEY Gold 8 AS "command."
* DEFINE KEY FUNCTION 34 AS "command."
```

You can define Gold-keyboard keys. A Gold-keyboard key refers to the combination of Gold with a key on the main keyboard.

```
* DEFINE KEY Gold A AS "command."
```

When you define a Gold-keyboard key sequence, enclose the following symbols within quotation marks (except for quotation marks, which you must enclose within apostrophes):

exclamation point (!)
percent sign (%)
apostrophe (')
quotation marks (")

For example, you can define the following Gold sequence to exit from EDT in one step, without having to press several keys and type EXIT:

```
* DEFINE KEY Gold "!" AS "EXT EXIT"
```

You can also define keys using the control (Ctrl) key with letter keys as well as with brackets ([ ]), a backslash (\ ), a tilde (~), a circumflex (^), and an underscore (_).

```
* DEFINE KEY CONTROL A AS "command."
* DEFINE KEY CONTROL ] AS "command."
```

You can also define Gold-Ctrl keys (for example, DEFINE KEY GOLD CONTROL AS *command*). When you use Gold and Ctrl, press the Gold key first, then hold down the Ctrl key while you press the keyboard key.

Do not define the following key combinations:

Ctrl/C
Ctrl/O
Ctrl/Q
Ctrl/S
Ctrl/T (unless you SET NOCONTROL=T)
Ctrl/X
Ctrl/Y
Gold-Ctrl/C
Gold-Ctrl/O
Gold-Ctrl/Q
Gold-Ctrl/S
Gold-Ctrl/X
Gold-Ctrl/Y

Gold-digit (keyboard keys 0 through 9)
Gold-minus sign (–)

You can redefine the following key combinations, but be aware of their default functions:

Ctrl/H (Backspace)
Ctrl/I (Tab)
Ctrl/J (Line Feed)
Ctrl/M (Return)

2.9.3 Saving Key Definitions

Once you define a key, it stays defined throughout your editing session until you enter EXIT or QUIT or redefine the key. You can save key definitions, however, by including them in your startup command file or in a macro file. The following is a sample section from a startup command file:

```
DEFINE KEY CONTROL R AS "EXT INCLUDE ?'INCLUDE FILE: '."❶
DEFINE KEY CONTROL G AS "EXT FIND=MAIN."❷
DEFINE KEY CONTROL L AS "EXT EXIT."❸
DEFINE KEY CONTROL N AS "EXT QUIT."❹
```

When you press:

❶ Ctrl/R, EDT prompts you for a file to include.

❷ Ctrl/G, EDT puts you at the top of the MAIN buffer.

❸ Ctrl/L, EDT exits your editing session.

❹ Ctrl/N, EDT quits your editing session.

See Section 2.11 for more information about saving key definitions in startup command files.

2.10 Using Macros

EDT allows you to define a series of line-mode commands as a macro. Section 2.10.1, Section 2.10.2, Section 2.10.3, Section 2.10.4, Section 2.10.5, and Section 2.10.6 explain how and why you would want to do this.

2.10.1 Introduction

A macro is a sequence of line-mode commands that you can execute when you type the macro name. You use the DEFINE MACRO command to add the macro name to the list of valid line-mode commands for the duration of your editing session.

2.10.2 Creating a Macro

To create a macro, assign a name—for example, CONCERTS—with the DEFINE MACRO command:

```
* DEFINE MACRO CONCERTS
```

Next, enter a buffer of the same name:

```
* FIND=CONCERTS
```

Then, type the list of line-mode commands; in this case, use the INSERT command to create a list of dates. Type the INSERT command followed by the semicolon, one space, and then the text to be inserted. End the list with Ctrl/Z to invoke the line-mode prompt. (To return to the current line in the MAIN buffer, enter the FIND=MAIN command.)

```
* INSERT
                INSERT; February 11, 1990
                INSERT; May 2, 1990
                INSERT; December 9, 1990
                INSERT; January 20, 1991
                INSERT; March 20, 1991
                INSERT; April 7, 1991
                INSERT; August 25, 1992
^Z
[EOB]
* FIND=MAIN
```

EDT executes that series of commands whenever you type the name of the macro as a line command, as shown in the following example:

```
* CONCERTS
* TYPE WHOLE
    1         February 11, 1990
    2         May 2, 1990
    3         December 9, 1990
    4         January 20, 1991
    5         March 20, 1991
    6         April 7, 1991
    7         August 25, 1992
```

2.10.3 Macro Functions

Although macros can contain only line-mode commands, they are able to perform a variety of functions, including the following:

- Macros can contain SET commands to tailor your editing session.
- Macros can contain a series of line-mode commands.
- Macros can contain a series of key definitions.
- Macros can define other macros or call up macros from external files.

2.10.4 Comparing Macros to Startup Command Files

Both macros and startup command files contain line-mode commands, but macros are more flexible. A startup command file (see Section 2.11) is executed once at the beginning of an editing session. But, you can create and access many different macros containing different sets of commands during one editing session just by typing the macro name.

2.10.5 Saving Macros

Macros can be created at any time during your editing session and can be used for the remainder of the session. Since they are located in buffers, they disappear as soon as you leave EDT. If you create a macro that you want to save for other EDT sessions, you can use the EDT command WRITE to put a copy of the macro in an external file. When you need that macro again, use the INCLUDE command to copy the macro into a buffer and then establish the macro name as a command with the DEFINE MACRO command.

The following example illustrates this process:

```
$  EDIT FUN.FUN
* DEFINE MACRO HEADING  ❶
* FIND=HEADING  ❷
* INSERT
INSERT;  Name:  ❸
INSERT;  Address:
INSERT;  Number:
INSERT;  Date:
^Z
[EOB]
* WRITE HEADING.MAC  ❹
DISK$:[HARTWELL]HEADING.MAC;1  4 lines
* EXIT
    .
    .
    .
$  EDIT WHY.NOT
* HEADING  ❺
  ^
Unrecognized command
* FIND=HEADING
* INCLUDE HEADING.MAC  ❻
* DEFINE MACRO HEADING  ❼
* FIND=MAIN  ❽
* HEADING  ❾
* TYPE WHOLE

    Name:
    Address:
    Number:
    Date:
```

❶ The DEFINE MACRO command makes HEADING a valid line-mode command.

❷ You enter a buffer named HEADING.

❸ You use four INSERT commands to create the text you want to use.

❹ The WRITE command puts a copy of the macro in an external file.

❺ In a new editing session, EDT does not recognize the macro named HEADING until you define the macro with the DEFINE MACRO command.

❻ The INCLUDE command brings the macro in from an outside file and puts it in the buffer named HEADING.

❼ The DEFINE MACRO command makes HEADING a valid line-mode command.

❽ You return to the MAIN buffer. (You must leave the buffer containing the macro before you use the macro as a line-mode command.)

❾ The HEADING command now functions as a line-mode command.

Note: Macros override line-mode commands. If you create a macro with the same name as an existing line-mode command, EDT performs the macro, not the line-mode command. As long as the macro is in effect, it overrides the default line-mode command. If you need to reestablish the default use of the line-mode command, use the CLEAR command to eliminate the buffer with the same name.

2.10.6 Including Specifiers in a Macro

When you use line-mode commands that take specifiers, you must include the appropriate specifier in the macro text. For example, if you create a macro that includes a substitute command, you must supply the strings for EDT to use with that command.

You cannot use the SUBSTITUTE command in a macro and expect to enter the strings after typing the macro name.

The same rule applies when you use a command like TYPE in your macro. If you want to have a command that automatically types the current line and the next 9 lines, you can create the macro XTYPE to be:

```
TYPE . THRU +9
```

You cannot, however, create a command that lets you choose the number of lines to type and allows you to enter the final digit from the terminal.

2.11 Startup Command Files

A startup command file contains EDT line-mode commands that are executed when you invoke EDT—before you receive control of the editor. You can use startup command files to customize your EDT sessions. You can create startup command files in either your current or main directory. Some of the line-mode commands that a startup command file might contain are:

- DEFINE KEY commands—To redefine the function invoked by a function key, a keypad key, or a control character while you are editing in keypad mode.

- DEFINE MACRO commands—To associate a name with a sequence of line editing commands stored in a text buffer. You can then invoke the sequence by entering the macro name in response to the line editing asterisk prompt.
- INCLUDE commands—To bring text from a file into a text buffer. You might use them to load macros into a buffer, or to fill a buffer with text that you often use.
- SET commands—To establish EDT operating parameters. For example, SET TAB establishes the increment for structured tabs, and SET MODE CHANGE invokes keypad mode.

When you begin an editing session, EDT automatically searches your current directory for a startup command file named EDTINI.EDT. EDT then processes the commands in that file before giving control to you. For example, if you create an EDTINI.EDT file and include the following line-mode commands, EDT automatically executes these commands the next time you invoke EDT:

```
$  EDIT EDTINI.EDT [Return] ❶
Input file does not exist
[EOB]
* INSERT [Return]
        SET QUIET [Return] ❷
        SET NONUMBERS [Return] ❸
[Ctrl/Z]
* EXIT
$  EDIT LIST.DAT ❹
```

❶ EDT is invoked to create the file named EDTINI.EDT.

❷ SET QUIET suppresses the bell sound when EDT prints an error message (in keypad mode).

❸ SET NONUMBERS suppresses line numbers in line mode.

❹ Now when you invoke EDT to edit a file named LIST.DAT, the commands in EDTINI.EDT execute before you gain control of the editor.

Because EDTINI.EDT is the default startup command file, you do not need to name it in the command line. If you want EDT to use a startup command file with another name, you must include that command file specification in the EDIT/EDT command line.

The following example demonstrates how to create a two-line startup command file:

```
$  EDIT SETUP.EDT
Input file does not exist
[EOB]
* INSERT
        SET MODE CHANGE
        SET LINES 5
[Ctrl/Z]
[EOB]
* EXIT
```

In this example, the name of your startup command file is SETUP.EDT. It contains two SET commands: SET MODE and SET LINES. SET MODE CHANGE causes EDT to begin your editing session in keypad mode, and SET LINES 5 limits the display of text on your screen to five lines.

Use the qualifier /COMMAND= with the EDIT command to indicate the startup command file you want EDT to use. The following command line tells EDT to execute the line-mode commands in the startup command file named SETUP.EDT to edit a file named FUN.FUN:

```
$  EDIT/COMMAND=SETUP.EDT FUN.FUN
```

The following startup command file contains two SET commands, three DEFINE KEY commands, and one DEFINE MACRO command:

```
SET WRAP 60 ❶
SET SEARCH EXACT ❷
DEFINE KEY Gold W AS "CHGUW." ❸
DEFINE KEY CONTROL B AS "EXT INCLUDE ?'INCLUDE FILE: '." ❹
DEFINE KEY CONTROL G AS "EXT FIND=MAIN." ❺
FIND=GENERAL ❻
INSERT ;SET SEARCH GENERAL ❼
DEFINE MACRO GENERAL ❽
FIND=MAIN ❾
```

❶ Limits the line length for inserting text in keypad mode to 60 characters.

❷ Causes EDT to match exactly the case and diacritical marks (for example, grave accent or circumflex) of letters in search strings.

❸ Defines Gold/W to change all lowercase letters in a word to uppercase.

❹ Defines Ctrl/B to extend the INCLUDE line-mode command to keypad mode and issue a prompt.

❺ Defines Ctrl/G to return to the MAIN buffer.

❻ Creates a buffer named GENERAL and moves there.

❼ Inserts the line-mode command SET SEARCH GENERAL.

❽ Adds GENERAL to the list of valid line-mode commands.

❾ Returns to the first line of the MAIN buffer.

3 Formatting Files with DSR

DIGITAL Standard Runoff (DSR) is a program that formats text. DSR commands and flags, which are entered into text, allow you to specify the final appearance of a document. Neither the DSR commands nor flags appear in your final document.

3.1 Introduction

You can use DSR to determine your page size, create justified or unjustified right margins, place space between lines, and form lists. DSR can center titles, supply page numbers, and organize your text into sections and chapters. You can also use DSR to create a table of contents and an index. For a complete list of all the available DSR commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.1.1 Using DSR Defaults

When you process a file using DSR, a certain set of DSR commands affect your file by default. You do not need to add these DSR commands to your file to achieve the desired effect because DSR does it automatically. If you do not want your file to be affected by these DSR default commands, you must disable them. See the *VAX DIGITAL Standard Runoff Reference Manual* for a complete list of the default commands provided by DSR and the commands you need to disable them. The most common of the defaults are discussed here.

When you use DSR to process a file, your output file looks different from your input file because DSR provides the following standard format defaults:

- A text width of 70 characters and a text length of 58 lines.
- Sequential page numbering (omitting a page number for the first page).
- A left margin setting of 0 and a right margin setting of 70.
- Single spacing.
- Tab stops at character positions 9, 17, 25, 33, and so on.
- Filling: DSR fills each line with as many words as possible until the addition of another complete word would exceed the right margin.
- Justification: DSR adds spaces between words to expand each line exactly to the right margin, making the right margin even, or justified.

For example, if you create a file containing the text in the next example and then process the file using DSR, the resulting output file will be 70 characters wide (beginning at character position 0 and ending at character position 70), single spaced, filled, and justified.

Input File:

```
Children learn to speak through imitation.

Researchers find that infants first make random sounds.
Naturally, their parents praise them for sounds that are
culturally meaningful, so children begin to
repeat a subset of sounds. In time, babies deliberately
imitate the sounds made by their parents.
As they grow, children expand their language beyond the
family.  They incorporate language learned
from friends, neighbors, television, radio, and books.
```

Output File:

```
Children learn to speak through imitation.  Researchers find that
infants first make random sounds.  Naturally, their parents praise
them for sounds that are culturally meaningful, so children begin to
repeat a subset of sounds.  In time, babies deliberately imitate the
sounds made by their parents.  As they grow, children expand their
language beyond the family.  They incorporate language learned from
friends, neighbors, television, radio, and books.
```

3.1.2 Including DSR Commands

To use DSR, you create a file, insert text, and include DSR commands in the appropriate places. DSR creates an output file with a file type of MEM (for example, TEXT.MEM). The MEM file contains the formatted text, which you can then print or display on your terminal.

Use the following steps to format text using DSR:

What You Do	How You Do It
Create a file	$ EDIT list.rno Input file does not exist [EOB] *
Insert text with DSR commands	.LIST .LIST ELEMENT;apples .LIST ELEMENT;plums .LIST ELEMENT;oranges .LIST ELEMENT;bananas .END LIST
Process file	$ RUNOFF list.rno
Display formatted file	$ Type LIST.MEM 1 apples 2 plums 3 oranges 4 bananas

3.1.3 Looking at DSR Commands

All DSR commands are English words preceded by a control flag, which is often a period (.), as in the following examples:

```
.BLANK
.CENTER
```

The .BLANK command inserts a blank line. The .CENTER command centers the text immediately following it on the same line. You can abbreviate all DSR commands. The abbreviations for .BLANK and .CENTER are .B and .C, respectively. For a complete list of valid DSR command abbreviations, see the *VAX DIGITAL Standard Runoff Reference Manual*.

You end a DSR command with a terminator. Commands are most commonly terminated by the end of a line, but you can also use a semicolon (;) as a terminator, or you can terminate a command and begin another one with a period. You can also combine terminators. For example, you can terminate a command with a semicolon and another command.

DSR commands can be typed in uppercase, lowercase, or a combination of uppercase and lowercase.

The following table shows four ways to terminate a DSR command:

DSR Command	Terminator
.BLANK2	(end of line)
.BLANK2;	semicolon
.BLANK2.CENTER;text	another command
.BLANK2;.CENTER;text	semicolon and another command

On any line containing a DSR command, the command must begin in column 1, unless it follows other commands on the same line. Depending on the particular command, a line containing a command can contain additional commands or text.

The following example demonstrates how to use the .BLANK and .CENTER commands to format text in a file called DIETING.RNO:

```
.CENTER;Twelve Days of Dieting ❶
.BLANK3 ❷
On the twelfth  day of dieting, Millitsa gave to me,
.BLANK ❸
Twelve hot fudge sundaes,
.BLANK
Eleven gooey doughnuts,
.BLANK
Ten cherry cheese cakes,
.BLANK2 ❹
Nine lady fingers,
.BLANK ❺
Eight date nut muffins,
.BLANK
Seven oatmeal cookies,
.BLANK
Six bags of corn chips,
.BLANK2
.CENTER;FIVE COFFEE RINGS, ❻
.BLANK3
Four sticky buns,
.BLANK
Three candy bars,
.BLANK
Two marbled cakes,
.BLANK
And a pizza with pepperoni.
```

❶ The .CENTER command terminated by a semicolon centers the title.

❷ The .BLANK command followed by the number 3 creates three blank lines.

❸ The .BLANK command creates one blank line.

❹ The .BLANK command followed by the number 2 creates two blank lines.

❺ The .BLANK command creates one blank line.

❻ The .CENTER command terminated by a semicolon centers the line of text.

When you enter the command RUNOFF DIETING.RNO, DSR formats your text to look like this:

```
                    Twelve Days of Dieting

On the twelfth day of dieting, Millitsa gave to me,
Twelve hot fudge sundaes,
Eleven gooey doughnuts,
Ten cherry cheese cakes,
```

```
Nine lady fingers,
Eight date nut muffins,
Seven oatmeal cookies,
Six bags of corn chips,
                                FIVE COFFEE RINGS,

Four sticky buns,
Three candy bars,
Two marbled cakes,
And a pizza with pepperoni.
```

3.1.4 Running DSR to Process Your Files

After you add DSR commands to your file and exit from the editor, you are ready to run DSR. To run DSR, you enter the RUNOFF command followed by the name of the file you want to process. For example, to process a file named FUN.FUN, enter the following command line:

```
$ RUNOFF FUN.FUN
```

If you process a file with a file type of RNO, you only need to enter the file name, not the file type. For example, to process a file named FUN.RNO, enter the following command line:

```
$ RUNOFF FUN
```

3.1.4.1 Using Qualifiers with the RUNOFF Command

DSR provides twenty different qualifiers that you can use with the RUNOFF command. These qualifiers are as follows:

/BACKSPACE
/BOLD
/CHANGE_BARS
/CONTENTS
/DEBUG
/DOWN
/FORM_SIZE
/INDEX
/LOG
/MESSAGES
/NONSPACING_UNDERLINE
/OUTPUT
/PAGES
/PAUSE
/RIGHT
/SEPARATE_UNDERLINE
/SEQUENCE
/SIMULATE
/UNDERLINE_CHARACTER
/VARIANT

Section 3.13 explains how to create a table of contents and an index by using the qualifiers /CONTENTS and /INDEX. All the qualifiers are discussed in detail in the *VAX DIGITAL Standard Runoff Reference Manual*. Section 3.1.4.2 discusses how to use one of the qualifiers, /MESSAGES.

3.1.4.2 Using the /MESSAGES Qualifier

If DSR finds any errors during processing, it prints an error message (both on the screen and in the MEM file) telling you what the error is and where it is located in both the input and the output files. You can then correct the RNO file (by using a text editor) and reprocess the file.

You can use the /MESSAGES qualifier to specify where you want DSR to display error messages. The options are the following:

OUTPUT	Sends error messages only to the output file (MEM file)
USER	Sends error messages only to the terminal (screen)

The default is /MESSAGES=(OUTPUT,USER), which sends messages to the output file and displays them on the terminal. You can prevent error messages from going either to the output file or to the terminal, but you cannot suppress them entirely.

The following command causes DSR to display error messages in the MEM file, not at the terminal:

```
$ RUNOFF FUN/MESSAGES=OUTPUT
```

The following command causes DSR to display error messages at the terminal, not in the MEM file:

```
$ RUNOFF FUN/MESSAGES=USER
```

3.1.5 Stripping MEM Files of Carriage-Return/Line-Feed Symbols

When you edit a MEM file produced by DSR, you will notice that the file contains carriage-return/line-feed symbols. The format of the symbols is dependent on the text editor you use. You can also have a file containing these symbols when you add a MEM file to an RNO file. In either case, you may want to strip the MEM file of these symbols using a text editor. Removing the symbols does not break your file. See the documentation for your text editor for instructions on taking carriage-return/line-feed symbols from the file.

3.2 Formatting Lists

You need to know only the following three DSR commands to format a numbered list:

- .LIST
- .LIST ELEMENT
- .END LIST

Begin your list with the .LIST command. The .LIST command causes DSR to start a list. When DSR formats a list, it indents the left margin, puts a blank line before the first list item, after the last item, and between each item in the list. DSR also numbers (or otherwise marks) the items.

Precede each item in your list by the .LIST ELEMENT command. Use a semicolon to separate the list item from the command or put the list item on the following line. End your list with the .END LIST command.

The following example shows how to format a list using DSR commands:

Input file (LIST.RNO)

```
.LIST
.LIST ELEMENT;grosbeak
.LIST ELEMENT;gold finch
.LIST ELEMENT;redpoll
.LIST ELEMENT;sparrow
.END LIST
```

Output file (LIST.MEM)

```
1.  grosbeak
2.  gold finch
3.  redpoll
4.  sparrow
```

The following sections describe other kinds of lists and the DSR commands you need to create them.

3.2.1 Creating Bulleted Lists

By default, DSR creates a numbered list. If you do not want your list to be numbered, you can substitute bullets or other characters for the numbers. For example, if you want a bulleted list, enter the .LIST command followed by a lowercase o, or bullet, in quotation marks ("o"):

```
.LIST "o"
```

The following example shows how to make a bulleted list:

Input file (LIST.RNO)

```
.LIST "o"
.LIST ELEMENT;ferret
.LIST ELEMENT;mink
.LIST ELEMENT;rabbit
.LIST ELEMENT;sable
.LIST ELEMENT;raccoon
.END LIST
```

Output file (LIST.MEM)

```
o  ferret
o  mink
o  rabbit
o  sable
o  raccoon
```

3.2.2 Creating Lists Using Any Symbol

If you do not want numbered or bulleted lists, you can choose another character or symbol and enclose it in quotation marks following the .LIST command. Asterisks (*), dollar signs ($), and at signs (@) are used in the following examples:

Input file (LIST.RNO)

```
.LIST "*"
.LIST ELEMENT;jupiter
.LIST ELEMENT;mars
.LIST ELEMENT;venus
.END LIST
```

Output file (LIST.MEM)

```
* jupiter
* mars
* venus
```

Input file (LIST.RNO)

```
.LIST "$"
.LIST ELEMENT;mark
.LIST ELEMENT;yen
.LIST ELEMENT;dollar
.LIST ELEMENT;pound
.END LIST
```

Output file (LIST.MEM)

```
$ mark
$ yen
$ dollar
$ pound
```

Input file (LIST.RNO)

```
.LIST "@"
.LIST ELEMENT;why
.LIST ELEMENT;not
.LIST ELEMENT;?
.END LIST
```

Output file (LIST.MEM)

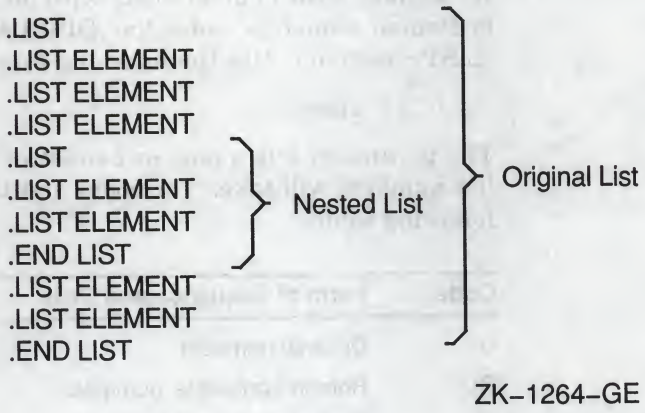
```
@ why
@ not
@ ?
```

3.2.3 Creating Nested Lists

You can also make a nested list—that is, a list indented within a list—by entering another .LIST command between the original .LIST and .END LIST commands. The new .LIST command temporarily suspends the characteristics of the original list. The next .END LIST command ends only the new .LIST command and restores the characteristics of the previous one. You must terminate each list with the .END LIST command.

Figure 3–1 displays the original list containing five elements and the nested list containing two elements.

Figure 3–1 Creating a Nested List



Notice that the nested list in the following example has bullets, unlike the original numbered list:

Input file (LIST.RNO)

```
.LIST
.LIST ELEMENT;German
.LIST ELEMENT;Russian
.LIST ELEMENT;Swedish
.LIST ELEMENT;Yugoslavian
.LIST "o"
.LIST ELEMENT;Serbian
.LIST ELEMENT;Croatian
.LIST ELEMENT;Macedonian
.END LIST
.LIST ELEMENT;Turkish
.LIST ELEMENT;Scottish
.LIST ELEMENT;Irish
.END LIST
```

Output file (LIST.MEM)

```
    1.  German

    2.  Russian

    3.  Swedish
```

```
4.  Yugoslavian
      o  Serbian
      o  Croatian
      o  Macedonian
5.  Turkish
6.  Scottish
7.  Irish
```

3.2.4 Creating Lists with Letters and Roman Numerals

By default, DSR numbers lists with decimal numbers. To list with letters or Roman numerals, enter the .DISPLAY ELEMENTS command after the .LIST command. Use the following syntax:

```
.DISPLAY ELEMENTS x
```

The parameter *x* is a one- or two-letter code that specifies the form the list numbers will take. The codes and their explanations are shown in the following table:

Code	Form of Sequence and Case
D	Decimal numbers
RU	Roman uppercase numerals
RL	Roman lowercase numerals
LU	Uppercase letters
LL	Lowercase letters

The following examples show lowercase Roman numerals and letters. Notice that the .DISPLAY ELEMENTS command appears between the .LIST command and the first .LIST ELEMENT command.

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;tan
.LIST ELEMENT;beige
.LIST ELEMENT;rust
.LIST ELEMENT;brown
.END LIST
```

Output file (LIST.MEM)

```
    i.  tan
   ii.  beige
  iii.  rust
   iv.  brown
```

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;January
.LIST ELEMENT;February
.LIST ELEMENT;March
.END LIST
```

Output file (LIST.MEM)

```
a.  January
b.  February
c.  March
```

In the following example, notice that each .END LIST command ends the sequence that you specified with .DISPLAY ELEMENTS. You must enter the .DISPLAY ELEMENTS command each time you start a list in which you want to change the numbers or letters. If you want each list to be marked by the same characters, you still must enter a .DISPLAY ELEMENTS command for every .LIST command.

Input file (LIST.RNO)

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;marble
.LIST ELEMENT;maple
.LIST ELEMENT;alabaster
.LIST ELEMENT;oak
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;light
.LIST ELEMENT;dark
.END LIST
.LIST ELEMENT;glass
.LIST ELEMENT;plastic
.LIST ELEMENT;mahogany
.END LIST
```

Output file (LIST.MEM)

```
 i.  marble
ii.  maple
iii. alabaster
 iv. oak
     a.  light
     b.  dark

 v.  glass
vi.  plastic
vii. mahogany
```

3.3 Formatting Memos

You can use the following DSR commands to format a memo:

- .BLANK
- .BREAK
- .TAB STOPS

You can organize these three DSR commands into a memo skeleton, like the following one, and fill in the variable information every time you need to send a memo. (Notice that you must press the Tab key to activate the .TAB STOPS command.)

Input file (SKELETON.RNO)

```
.TAB STOPS 30
Tab   DATE:
.BREAK
Tab   FROM:
.BREAK
Tab   DEPT:
.BREAK
Tab   EXT:
.BREAK
Tab   LOC:
.BLANK 2
TO:
.BLANK
SUBJECT:
.BLANK 2
(Enter text of memo here.)
```

Output file (SKELETON.MEM)

```
                                DATE:
                                FROM:
                                DEPT:
                                EXT:
                                LOC:

TO:

SUBJECT:

(Enter text of memo here.)
```

The .BREAK command ends the current line immediately, without filling or justification. The .TAB STOPS *n* command changes the current position of the tab stops. The value of *n* specifies the character position for a tab stop.

The next memo example uses the following DSR commands:

```
.BLANK
.BREAK
.LEFT MARGIN
.LIST
.LIST ELEMENT
.END LIST
.LITERAL
.END LITERAL
.TAB STOPS
```

Input file (SPY.RNO)

```
.TAB STOPS 40 ❶

[Tab]DATE: 19-Apr-90

.BREAK ❷

[Tab]FROM: A. Gent

.BREAK

[Tab]DEPT: Topspy

.BREAK

[Tab]EXT: 007

.BREAK

[Tab]LOC: Swiss Branch

.BLANK

TO: Underlings

.BLANK

SUBJECT: New Equipment
.BLANK 2

The following items will be discussed in our
next meeting:

.LIST ❸
.LIST ELEMENT;super slim tie tack monitor
.LIST ELEMENT;wired socks
.LIST ELEMENT;mini diamond stud earring camera
.END LIST

Please attempt to decode the following message
for your next assignment:

.LITERAL ❹
                    que estap dyi

                         fho
                              syto
                  szru
                          fsohg
                        wquip
                     dwd

.END LITERAL ❺
```

❶ The .TAB STOPS 40 command sets the first tab stop to character position 40.

❷ The .BREAK command ends the current line immediately and starts a new line without adding a blank line.

❸ The .LIST command begins a list.

❹ The .LITERAL command displays the text literally, exactly as it is entered.

❺ The .END LITERAL command stops displaying the text literally.

Output file (SPY.MEM)

```
                                        DATE: 13-Nov-88
                                        FROM: A.  Gent
                                        DEPT: Topspy
                                        EXT: 007
                                        LOC: Swiss Branch

TO: Underlings

SUBJECT: New Equipment

The following items will be discussed in our next meeting:

     1.  super slim tie tack monitor

     2.  wired socks

     3.  mini diamond stud earring camera

Please attempt to decode the following message for your next
assignment:

                que estap dyi

                    fho
                         syto
            szru
                      fsohg
                     wquip
                  dwd
```

3.4 Filling and Justifying Text

By default, DSR automatically fills and justifies text (.FILL and .JUSTIFY commands) when you process a file. The .FILL command causes DSR to fill each line with as many words as possible until the addition of another word would exceed the right margin. The .JUSTIFY command causes DSR to add spaces between words to expand each line exactly to the right margin, making the margin even, or justified. If you do not want DSR to fill or justify the text in your file, you must disable these commands by entering the .NO FILL and .NO JUSTIFY commands.

3.4.1 DSR Commands .FILL and .JUSTIFY (defaults)

The following example demonstrates how DSR fills and justifies text by default:

Input file (VERSE.RNO)

```
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out, That what  we  have
we prize not to the worth while we enjoy
it; but being lacked, lacked  and  lost,
Why, then we rack the value.
```

3.4.2 DSR Commands .NO FILL and .JUSTIFY

When you do not want each line of your text to fill with words, enter the .NO FILL command. The .NO FILL command also turns off justification. To obtain an even right margin, include the .JUSTIFY command after the .NO FILL command. Entering the .NO FILL command with the .JUSTIFY command produces the following variation of the original example. Notice how the .JUSTIFY command spaces the words on each line to create an even right margin.

Input file (VERSE.RNO)

```
.NO FILL
.JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For      it      so      falls      out,
That        what         we         have
we               prize                not
to               the                worth
while         we        enjoy         it;
but             being              lacked,
lacked             and               lost,
Why,
then     we     rack     the      value.
```

3.4.3 DSR Commands .FILL and .NO JUSTIFY

When you want each line of text to fill with words but do not want an even right margin, use the .FILL and .NO JUSTIFY commands. Notice the ragged right margins in the following example:

Input file (VERSE.RNO)

```
.FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out, That what we have
we prize not to the worth while we enjoy
it; but being lacked, lacked and lost,
Why, then we rack the value.
```

3.4.4 DSR Commands .NO FILL and .NO JUSTIFY

When you want your text to stay the way you type it, that is, without filling and justification, enter the .NO FILL and .NO JUSTIFY commands.

Input file (VERSE.RNO)

```
.NO FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

Output file (VERSE.MEM)

```
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

3.5 Adjusting the Text Display

By default, DSR provides a page length of 58 lines and a page width of 70 characters. When you want to change the size of your page, use the .PAGE SIZE and .RIGHT MARGIN commands. The .PAGE SIZE command specifies the number of lines of text on a page and the page width for the running heads. The syntax for these commands is as follows:

```
.PAGE SIZE n
.RIGHT MARGIN n
```

The parameter *n* with the .PAGE SIZE command indicates the maximum number of lines on a page. It cannot be smaller than 13. The parameter *n* with the .RIGHT MARGIN command indicates the maximum number of characters on a line. It cannot be larger than 150.

The following example shows how to create two different page sizes by entering the .PAGE SIZE command along with the .RIGHT MARGIN command:

Input file (PAGESIZE.RNO)

```
.PAGE SIZE 15
.RIGHT MARGIN 20
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
```

Output file (PAGESIZE.MEM)

```
page size page  size
15   lines  long  20
characters wide page
size  page  size  15
lines    long     20
characters wide page
size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide page

                Page 2


size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide
```

Input file (PAGE.RNO)

```
.PAGE SIZE 25
.RIGHT MARGIN 35
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
```

Output file (PAGE.MEM)

```
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide  page  size page size 25 lines
long 35 characters wide  page  size
page   size   25   lines   long  35
                                Page 2

characters wide page size page size
25  lines  long  35 characters wide
page size page size 25  lines  long
35  characters  wide page size page
size 25 lines  long  35  characters
wide
```

3.5.1 Indenting Text

By default, when you use the .INDENT command, DSR indents by five spaces the first line of text following the command. The syntax for the indent command is as follows:

```
.INDENT n
```

The parameter *n* specifies the number of character positions to the right of the .LEFT MARGIN setting that the line of text is indented. When you specify a negative number (*-n*), DSR moves the line of text *n* number of character positions to the left of the .LEFT MARGIN setting.

The following example demonstrates how to use the .INDENT command:

Input file (INDENT.RNO)

```
.LEFT MARGIN 10
.RIGHT MARGIN 60
The left margin setting for this text is 10 and the right margin
setting is 60.  If you want to indent text, use the .INDENT command.
.BLANK
.INDENT 20
This line of text is indented 20 spaces.
Every line you type from now on will be flush with the left
margin until you enter the .INDENT command again.
.BLANK
.INDENT 10
This line of text is indented 10 spaces.
If you want a line of text to begin to the left of the
left margin setting, specify a negative number (-n) with
the .INDENT command.
.BLANK
.INDENT -10
This line of text is 10 spaces to the left of the left margin.
This line of text begins at the left margin setting.  Any text
you enter at this point will remain flush with the left
margin until you enter the .INDENT command again.  If you
want to indent two or three lines of text, you must enter
the .INDENT command on each line you want to indent.
.BLANK
.INDENT 15
This is the first of three indented lines.
.INDENT 15
This is the second of three indented lines.
.INDENT 15
This is the third.
This line of text is back at the left margin again.
```

Output file (INDENT.MEM)

```
The left margin setting for this text  is  10  and
the  right  margin  setting is 60.  If you want to
indent text, use the .INDENT command.
                        This line of text is  indented
20  spaces.   Every line you type from now on will
be flush with the left margin until you enter  the
.INDENT command again.
          This line of text is indented 10 spaces.
If you want a line of text to begin to the left of
the left margin setting, specify a negative number
(-n) with the .INDENT command.
```

```
This line of text is 10 spaces to the left of the left
        margin.  This line of text begins at the left
        margin setting.  Any text you enter at this point
        will remain flush with the left margin until you
        enter the .INDENT command again.  If you want to
        indent two or three lines of text, you must enter
        the .INDENT command on each line you want to
        indent.
                        This is the first of three indented
        lines.
                        This is the second of three
        indented lines.
                        This is the third.  This line of
        text is back at the left margin again.
```

3.5.2 Placing a Single Line of Text Relative to the Right Margin

You can use the .RIGHT command to position a single line of text relative to the right margin. The syntax is as follows:

```
.RIGHT n;text
```

The number substituted for the parameter *n* can be either positive or negative. A positive number specifies how many character positions to the left of the right margin setting the line is indented. A negative number specifies the number of character positions to the right of the right margin setting that the line extends to. The parameter *text* indicates the text to be positioned relative to the right margin. No other DSR command can follow this text on a line.

The following example shows the .RIGHT command being used to notate various directions in the script of a play. (A positive number is specified for *n*.)

Input file (PLAY.RNO)

```
.LEFT MARGIN 0
.RIGHT MARGIN 50
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT 25;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT 25;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT 25;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT 25;(open door)
```

Output file (PLAY.MEM)

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                              (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
                        (shake head)
all you can think about is yourself.
Celia:  Maybe I could apologize...
                              (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
                        (open door)
```

The following example is identical to the previous example, except that a negative number is specified for *n*:

Input file (PLAY.RNO)

```
.LEFT MARGIN 0
.RIGHT MARGIN 50
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT -5;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT -5;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT -5;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT -5;(open door)
```

Output file (PLAY.MEM)

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                                                  (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
                                            (shake head)
all you can think about is yourself.
Celia:  Maybe I could apologize...
                                                  (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
```

3.6 Creating Space on a Page

If you want an example or figure to accompany your text, you need to create some space on the page for it. You can use several different DSR commands to create that necessary space. Some of these commands are as follows:

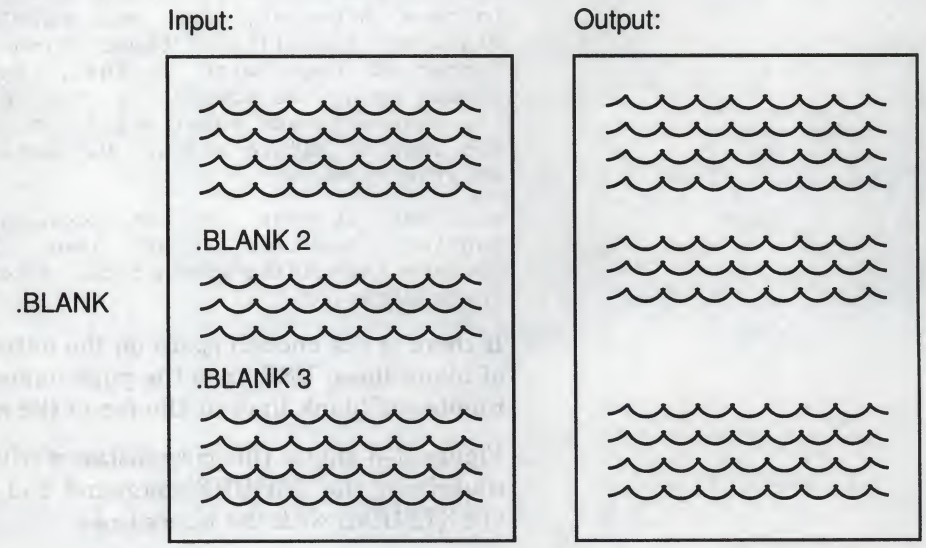
- .BLANK
- .FIGURE
- .FIGURE DEFERRED
- .LITERAL

3.6.1 Separating Sections with Blank Lines

If you use the .BLANK command, DSR leaves the specified number of lines blank. For example, .BLANK 2 produces two blank lines and .BLANK 10 produces ten blank lines. By default, .BLANK produces one blank line. The .BLANK command is useful for creating one or two blank lines between sections of text. (.BLANK does not work correctly when you use it at the top of a page.) If you enter the .BLANK 10 command near the bottom of the page, there may not be enough room on the page for all ten lines, resulting in your space being split between pages. To avoid possible space splits, use either the .FIGURE command or the .FIGURE DEFERRED command.

Figure 3–2 shows an input file (TEXT.RNO) containing the .BLANK command and the resulting output file (TEXT.MEM) with the blank line.

Figure 3–2 Using the .BLANK Command



ZK–1265–GE

3.6.2 Creating Uninterrupted Space

The .FIGURE command leaves space for the specified number of blank lines on the output page at the point in the input file where the command is entered. In the following example, the command .FIGURE 5 adds enough space for five lines after the words "entitle them":

Input file DECLARATION.RNO

```
.PAGE SIZE 25
.RIGHT MARGIN 40
When in the Course of human events it becomes necessary
for one people to dissolve the political bands which
have connected them with another, and to assume among
the powers of the earth, the separate and equal station
to which the Laws of Nature and of Nature's God entitle
them,
.FIGURE 5
a decent respect to the opinions of mankind
requires that they should declare the causes which
impel them to the separation.
```

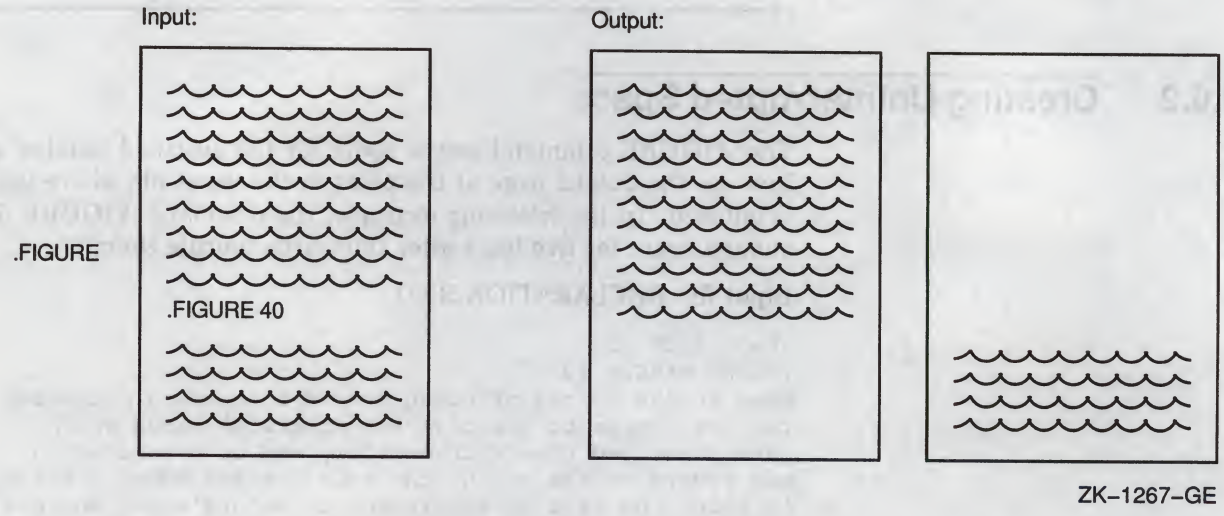
Output file DECLARATION.MEM

```
When in the Course of human events it
becomes necessary for one people to
dissolve the political bands which have
connected them with another, and to
assume among the powers of the earth,
the separate and equal station to which
the Laws of Nature and of Nature's God
entitle them,

a decent respect to the opinions of
mankind requires that they should
declare the causes which impel them to
the separation.
```

If there is not enough space on the output page for the specified number of blank lines, DSR ends the page immediately and puts the requested number of blank lines at the top of the next page.

Figure 3–3 shows this circumstance with an input file (TEXT.RNO) containing the .FIGURE command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–3 Using the .FIGURE Command

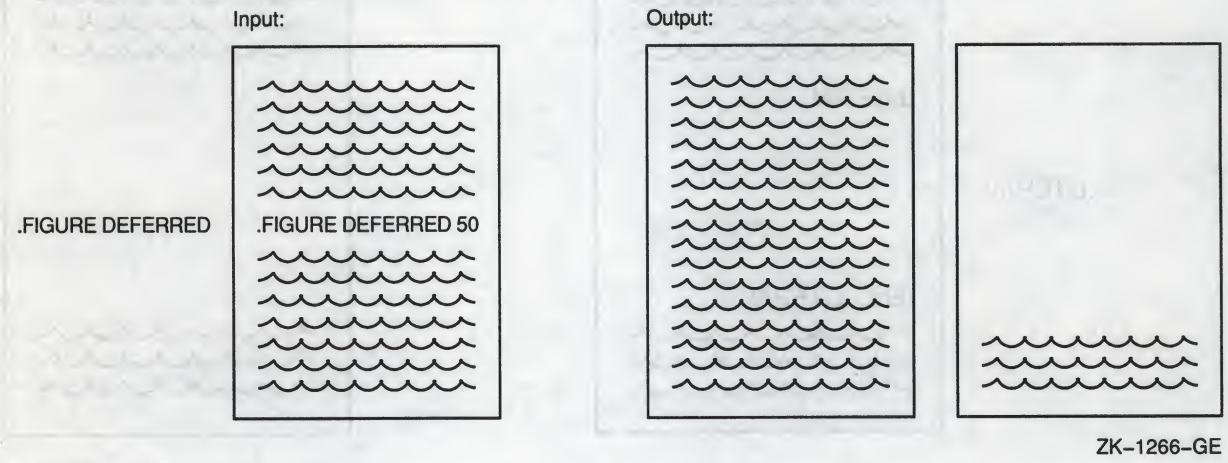


The .FIGURE DEFERRED command leaves enough space on the current page for the specified number of lines. If all the lines do not fit, .FIGURE DEFERRED fills the current page with text and creates space for the specified number of lines at the top of the next page.

For example, if you are about to add a 40-line figure to your file (using the .FIGURE 40 command) but you realize only 25 lines are left on the current output page, add the .FIGURE DEFERRED 40 command to the file instead of the .FIGURE 40 command. This causes DSR to fill the current output page with text and leave a 40-line space at the top of the next page for your figure.

Figure 3–4 shows an input file (TEXT.RNO) containing the .FIGURE DEFERRED command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–4 Using the .FIGURE DEFERRED Command

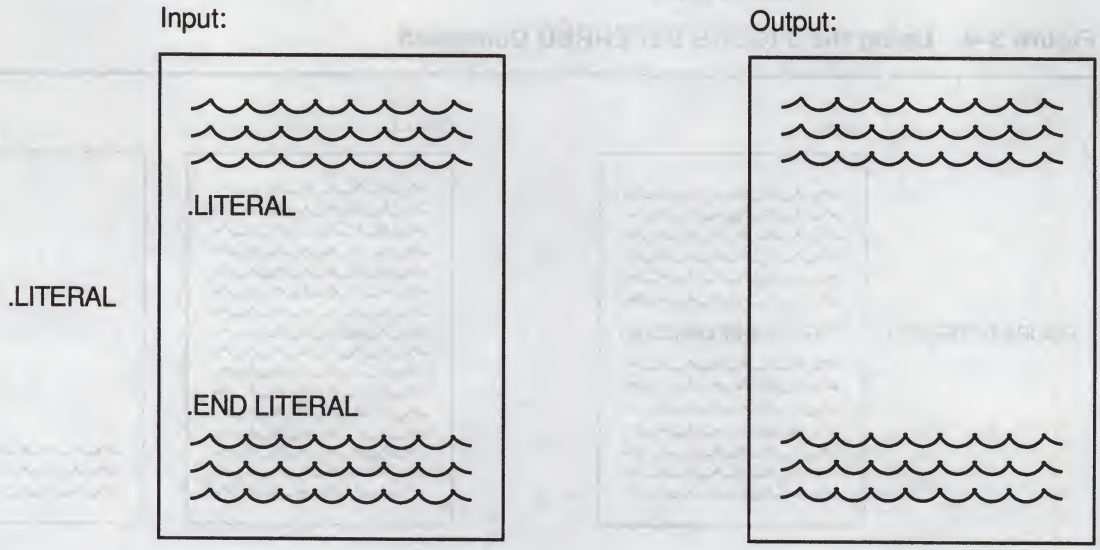


3.6.3 Seeing the Space You Create

You can also use the .LITERAL command to create space. If you want to see the amount of space you are creating, enter the .LITERAL command and press Return until the screen has enough blank space. Then, enter the .END LITERAL command. You might want to use the .LITERAL command if you have a picture to paste directly onto your text and want to see how it looks with the blank space you create.

Figure 3–5 shows an input file (TEXT.RNO) containing the .LITERAL command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 3–5 Using the .LITERAL Command



ZK–1268–GE

3.6.4 Example of Creating Space

The following example uses the .BLANK, .FIGURE DEFERRED, .FIGURE, and .LITERAL commands in a single file to show the different output that each produces:

Input file (SPACE.RNO)

```
.page size 20
.right margin 40
This is the first line on the page.
Each page will be 20 lines long and 40
characters wide.  If you enter
the .BLANK 2 command here, DSR
creates two blank lines.
.BLANK 2
There is not enough space on this page
for a figure that is 15 lines long, so
if you enter the .FIGURE DEFERRED 15
command here,
.FIGURE DEFERRED 15
DSR will fill up the rest of this page
with text, then create 15 lines of
space at the top of the next page.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,
.FIGURE 10
```

```
DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
But, if you enter the .FIGURE 10 command again,
and there is not enough space left on this
page for the entire figure (in this case 10),
DSR stops displaying text immediately
.FIGURE 10
and moves to the top of the next page, creating
the specified space (10 lines).  Then DSR
resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and press
Return three times,
.LITERAL


.END LITERAL
you will create three blank lines.  Remember to
enter the .END LITERAL command when you are
done.
```

Output file (SPACE.MEM)

```
This is the  first  line  on  the  page.
Each  page  will be 20 lines long and 40
characters  wide.   If  you  enter   the
.BLANK  2  command here, DSR creates two
blank lines.

There is not enough space on  this  page
for  a  figure that is 15 lines long, so
if you enter  the  .FIGURE  DEFERRED  15
command here, DSR will fill up the  rest
of  this  page with text, then create 15
lines  of  space   at   the   top   of   the
next page.  texttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
                                    Page 2
```

```
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
```

Page 3

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,

DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext

Page 4

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
But, if you enter the .FIGURE 10 command again, and there is not enough space left on this page for the entire figure (in this case 10), DSR stops displaying text immediately

Page 5

and moves to the top of the next page, creating the specified space (10 lines). Then DSR resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and

Page 6

press Return three times,

you will create three blank lines. Remember to enter the .END LITERAL command when you are done.

3.7 Formatting Sections

When you want to organize text into sections, use the .HEADER LEVEL command. The syntax for the command is as follows:

```
.HEADER LEVEL n title
```

The value of *n* indicates the specific header level. For example, .HEADER LEVEL 1 creates a section beginning with 1. If you enter a series of .HEADER LEVEL 1 commands, DSR produces section numbers starting with 1 and increasing by 1, as in the following example:

Input file (HEAD1.RNO)

```
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
```

Output file (HEAD1.MEM)

```
1

2

3

4

5
```

The .HEADER LEVEL 2 command affects the second digit. If you enter a series of .HEADER LEVEL 2 commands, the second digit changes, as in the following example:

Input file (HEAD2.RNO)

```
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
```

Output file (HEAD2.MEM)

```
0.1
0.2

0.3
0.4
```

The .HEADER LEVEL 3 command affects the third digit. An example of a series of .HEADER LEVEL 3 commands follows:

Input file (HEAD3.RNO)

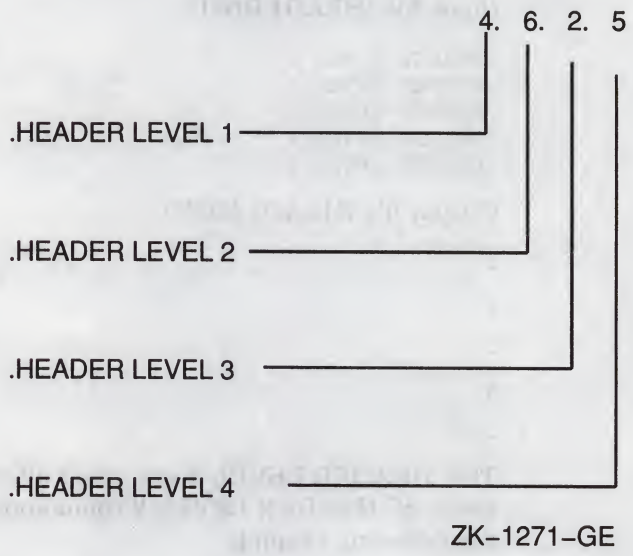
```
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
```

Output file (HEAD3.MEM)

```
0.0.1
0.0.2
0.0.3
0.0.4
```

Figure 3–6 shows one section number (4.6.2.5) and the various header levels corresponding to each part of the number.

Figure 3–6 Looking at Header Levels



3.7.1 Specifying a Title

You can follow the .HEADER LEVEL *n* command with a title. DSR uses uppercase for all the letters in a title of header level 1. A title of header level 2 has only initial caps. Header level 3 titles have initial caps followed by a hyphen to separate the title from the text that follows on the same line.

The following example contains a series of .HEADER LEVEL commands:

Input file (LEVEL.RNO)

```
.HEADER LEVEL 1this is a header level 1 title
.HEADER LEVEL 2this is a header level 2 title
.HEADER LEVEL 3this is a header level 3 title
This is the first line of text.
.HEADER LEVEL 1this is the next header level 1 title
```

Output file (LEVEL.MEM)

```
1  THIS IS A HEADER LEVEL 1 TITLE

1.1  This Is A Header Level 2 Title

1.1.1  This Is A Header Level 3 Title - This is the first line
of text.


2  THIS IS THE NEXT HEADER LEVEL 1 TITLE
```

The following example demonstrates how to organize recipes using the .HEADER LEVEL command:

Input file (RECIPES.RNO)

```
.HEADER LEVEL 1pies
An introduction to pies goes here.
.HEADER LEVEL 2grasshopper pie
The origin of the Grasshopper Pie goes here.
.HEADER LEVEL3chocolate wafer crust
This section explains how to make the crust.
.HEADER LEVEL3filling
This section explains how to make the filling.
.HEADER LEVEL2lemon meringue pie
Various lemon pies are discussed here.
.HEADER LEVEL3flaky pie crust
The recipe for a good, flaky crust goes here.
.HEADER LEVEL3lemon filling
This section describes how to make the filling.
.HEADER LEVEL3meringue topping
Meringues are mentioned here.
.HEADER LEVEL1layered cakes
Layered cakes are described here.
.HEADER LEVEL2millie's magnificent torte
Tortes are described here.
.HEADER LEVEL3walnut layers
A recipe for the walnut layers goes here.
.HEADER LEVEL3coffee frosting
Coffee frosting is explained here.
```

Output file (RECIPES.MEM)

```
1  PIES

An introduction to pies.goes here.

1.1  Grasshopper Pie

The origin of the Grasshopper Pie goes here.

1.1.1  Chocolate Wafer Crust - This section explains how  to
make the crust.

1.1.2  Filling - This  section  explains  how  to  make  the
filling.

1.2  Lemon Meringue Pie

Various lemon pies are discussed here.

1.2.1  Flaky Pie Crust - The recipe for a good, flaky  crust
goes here.

1.2.2  Lemon Filling - This section describes  how  to  make
the filling.
```

```
1.2.3  Meringue Topping - Meringues are mentioned here.

2  LAYERED CAKES

Layered cakes are described here.

2.2  Millie's Magnificent Torte

Tortes are described here.

2.2.1  Walnut Layers - A recipe for the walnut  layers  goes
here.

2.2.2  Coffee Frosting - Coffee frosting is explained here.
```

3.7.2 Using Roman Numerals or Letters

By default, DSR displays decimal numbers when you use the .HEADER LEVEL command. If you want to specify Roman numerals or letters, use the .DISPLAY LEVELS command. The syntax for this command follows:

```
.DISPLAY LEVELS y1,y2,y3...y6
```

The letter *y* is a one- or two-letter code. Some available codes follow:

Code	Form of Sequence and Case
RU	Roman uppercase numerals
RL	Roman lowercase numerals
LU	Uppercase letters
LL	Lowercase letters

For more information and a complete list of available codes, see the *VAX DIGITAL Standard Runoff Reference Manual.*

If you want the numbers in the first level heads to be uppercase Roman numerals, the second level to be uppercase letters, and the third level to be lowercase Roman numerals, enter the .DISPLAY LEVELS RU,LU,RL command. The resulting sections look as follows:

```
I  ALL HEADER LEVEL 1 TITLES ARE UPPERCASED

The .HEADER LEVEL 1 command produced this title.  The  code
RU  tells  DSR  to  make  all  level 1 heads uppercase Roman
numerals.

I.A  Header Level 2 Titles Have Initial Uppercase Letters

The .HEADER LEVEL 2 command produced this title.  The  code
LU tells DSR to make all level 2 heads uppercase letters.

I.A.i  Header Level 3 Titles Are Followed By A  Hyphen - The
.HEADER  LEVEL  3  command produced this title.  The code RL
tells  DSR  to  make  all  level  3  heads  lowercase  Roman
numerals.
```

3.8 Formatting Chapters

When you want to organize text into chapters, use the .CHAPTER command. The .CHAPTER command specifies the beginning of a chapter, numbers it, and allows you to supply a title. The syntax for the command is as follows:

```
.CHAPTER title
```

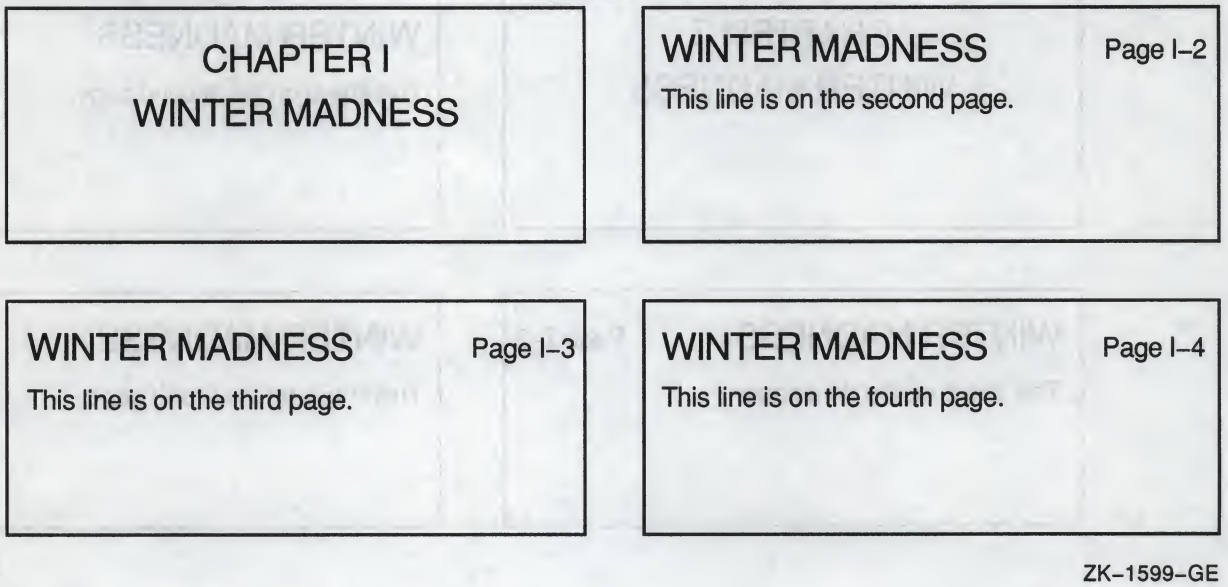
"Winter Madness" is the chapter title in the following example:

Input file (CHAP.RNO)

```
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
This line is on the third page.
.BLANK 20
This line is on the fourth page.
```

Figure 3–7 shows the output file (CHAP.MEM).

Figure 3–7 Using the .CHAPTER Command



ZK–1599–GE

3.8.1 Numbering Chapters

By default, the .CHAPTER command produces decimal chapter numbers. If you want Roman numerals or letters instead of decimal numbers, enter the .DISPLAY CHAPTER command before the .CHAPTER command you want to affect. The syntax for this command follows:

```
.DISPLAY CHAPTER y
```

The parameter *y* is one of the codes discussed in Section 3.7.2.

For more information about all the available codes, see the *VAX DIGITAL Standard Runoff Reference Manual*.

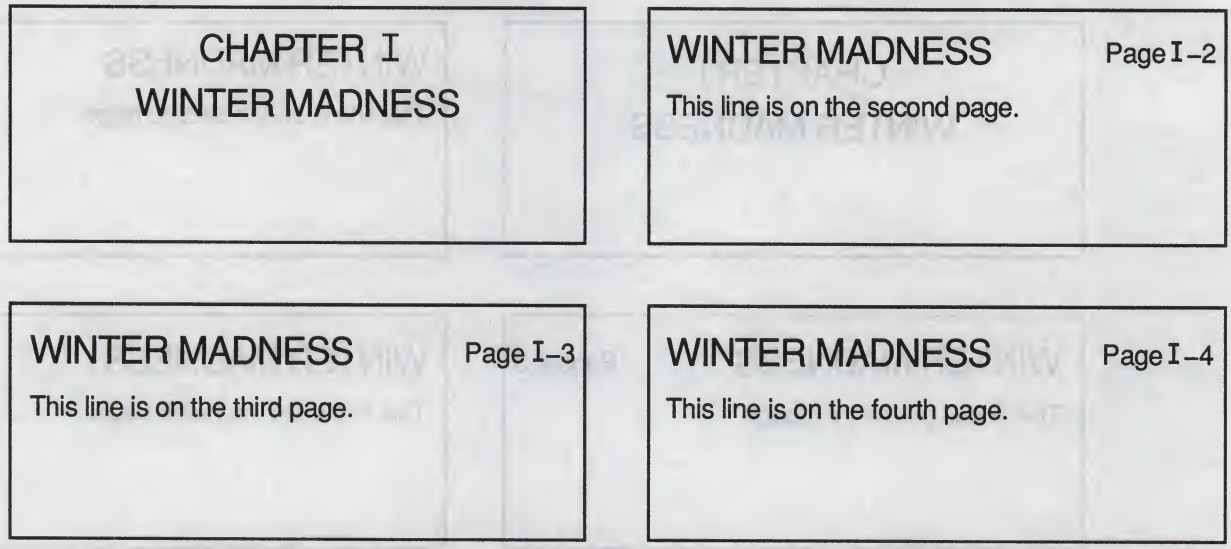
The following example shows how to create chapters numbered with uppercase Roman numerals by entering the .DISPLAY CHAPTER command with the code RU. Notice the uppercase Roman numeral page numbers.

Input file (CHAP.RNO)

```
.DISPLAY CHAPTER RU
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
This line is on the third page.
.BLANK 20
This line is on the fourth page.
```

Figure 3–8 shows the output file (CHAP.MEM).

Figure 3–8 Using the .DISPLAY CHAPTER Command



ZK–1598–GE

3.8.2 Changing the Way Pages Are Numbered

By default, DSR numbers pages using decimal numbers. If you want your pages lettered or numbered with Roman numerals, use the .DISPLAY NUMBER command before the page you want to affect. The syntax for this command follows:

```
.DISPLAY NUMBER y
```

The parameter *y* is one of the codes discussed in Section 3.7.2.

The following example shows the commands you enter to specify chapter and page numbers:

Input file (CHAP.RNO)

```
.DISPLAY CHAPTER RU ❶
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 20
.DISPLAY NUMBER LL ❷
This line is on the third page.
.BLANK 20
.DISPLAY NUMBER RL ❸
This line is on the fourth page.
```

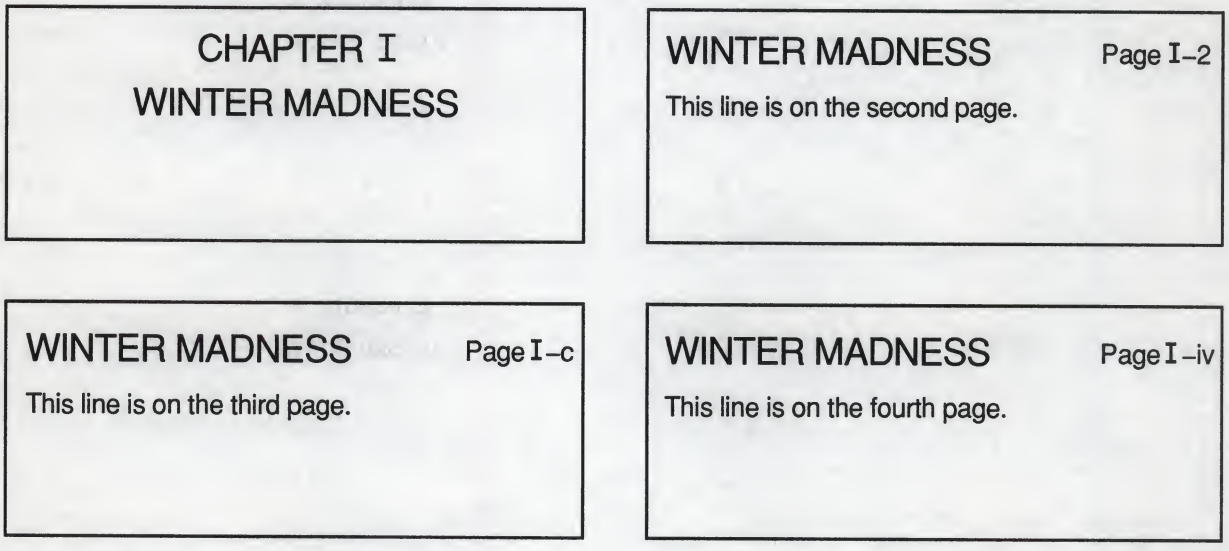
❶ This command numbers chapters with uppercase Roman numerals (RU).

❷ This command changes page numbers from decimal numbers to lowercase letters (LL).

❸ This command changes page numbers from lowercase letters (LL) to lowercase Roman numerals (RL).

Figure 3–9 shows the output file (CHAP.MEM).

Figure 3–9 Using the .DISPLAY NUMBER Command



ZK–1597–GE

3.9 Creating an Appendix

You can use the .APPENDIX command to specify the beginning of an appendix. DSR assigns an identifying letter to it and allows you to supply a title. Successive .APPENDIX commands assign identifying letters in alphabetical order.

The syntax for the .APPENDIX command is as follows:

```
.APPENDIX text
```

The variable *text* indicates the title you give the appendix.

The following example shows how to create three consecutive appendixes:

Input file (APPENDIX.RNO)

```
This is the last line of text before the appendixes.
.APPENDIX First Title
.APPENDIX Second Title
.APPENDIX Third Title
```

Output file (APPENDIX.MEM)

```
This is the last line of text before the appendixes.


                                APPENDIX A

                               FIRST TITLE









                                APPENDIX B

                              SECOND TITLE









                                APPENDIX C

                               THIRD TITLE
```

3.10 Creating Running Heads

By default, DSR provides page numbers at the top right of every page except the first. The .HEADERS ON command controls DSR's ability to create running page numbers. Running page numbers, which are provided by default, are just part of the information that DSR is able to display about the contents of a page. By using the DSR commands discussed in the following sections, you can create one or two lines of information (running heads) at the top of each page.

If you do not want page numbers, you can disable the page numbering default with the .NO NUMBER command.

The following example shows how DSR formats pages, providing running page numbers by default:

Input file (RUNNING.RNO)

```
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word "Page" at the top
right.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number.

                                                        Page 2

This line of text is on  the  second page.  From  this  point
on, every  page  has a number  preceded by the word "Page" at
the top right.

                                                        Page 3

This line of text is on the third page.  Notice the position
of the page number.
```

3.10.1 Specifying a Title

If you want your running head information to contain a title, use the .TITLE command. By default, DSR displays this title at the top left of every page except the first.

The following example shows how DSR formats pages when you enter the .TITLE command with running page numbers and a running title:

Input file (RUNNING.RNO)

```
.TITLE Whispering Willows
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number or a title.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word "Page"
at the top right and a title at the top left.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number and the title.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number or a title.

Whispering Willows                                      Page 2

This line of  text is on the second page.  From  this  point
on, every  page  has a number  preceded by the word "Page" at
the top right and a title at the top left.

Whispering Willows                                      Page 3

This line of text is on the third page.  Notice the position
of the page number and the title.
```

3.10.2 Specifying the Date

When you want the current date to appear in running heads, use the .DATE command. The date appears in the format dd mm yy, for example, 19 April 1990, on the right side of the subtitle line. The .SUBTITLE command must be included for the .DATE command to work.

The following example shows how DSR formats pages when you enter the .DATE command with the .TITLE command:

Input file (RUNNING.RNO)

```
.TITLE Whispering Willows
.SUBTITLE
.DATE
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word "Page" at the top
right, a title at the top left, and the date.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

Output file (RUNNING.MEM)

```
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.

Whispering Willows                                    Page 2
                                                 19 April 1990

This line of text is on the  second page.  From  this  point
on, every  page has a number preceded by the word  "Page" at
the top right, a title at the top left, and the date.

Whispering Willows                                    Page 3
                                                 19 April 1990

This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

3.10.3 Specifying a Subtitle

To specify a subtitle, enter the .SUBTITLE command after the .TITLE command.

The following example shows how to enter the .TITLE command and the .SUBTITLE command to create running heads with a title and a subtitle:

Input file (RUNNING.RNO)

```
.TITLE Country Gardens
.SUBTITLE Fragrant Garden Lilies
.PAGE SIZE 15
.BLANK 15
The first page does not have any running head information.
.BLANK 15
The title and subtitle appear at the top left of the second page.
The page number is at the top right.
.BLANK 15
Notice the position of the title, subtitle, and page
number on this page.
```

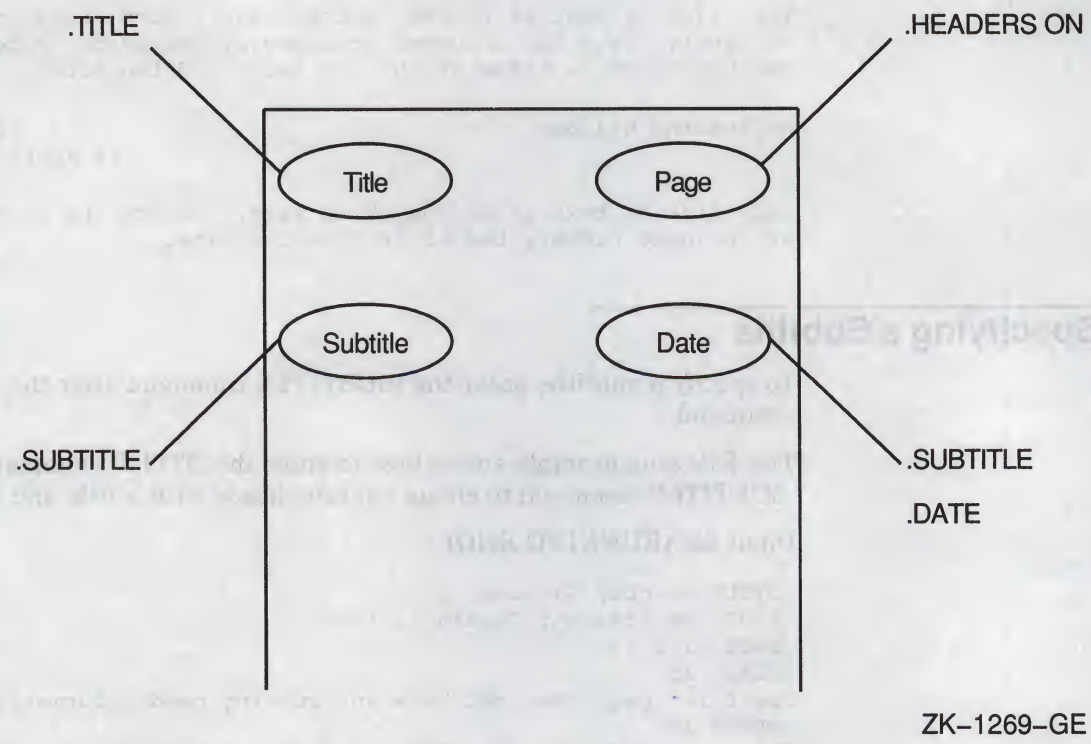
Output file (RUNNING.MEM)

```
The first page does not have any running head information.

Country Gardens                                       Page 2
Fragrant Garden Lilies

The title and subtitle appear at the top left of the  second
page.  The page number is at the top right.

Country Gardens                                       Page 3
Fragrant Garden Lilies

Notice the position of the title, subtitle, and page  number
on this page.
```

3.10.4 Organizing Running Head Information

Figure 3–10 shows the different parts of running head information and the commands you use to generate each part.

Figure 3–10 Running Head Information



You can use any combination of running head commands to alter the organization of information at the top of your documents.

3.10.5 Reorganizing Running Head Information

You can change the position of running head information on a page by using the .LAYOUT command. The .LAYOUT command can be entered before or after the running head information (.TITLE, .SUBTITLE, .DATE, .PAGE commands). You can center titles and subtitles at the tops of pages, center page numbers at the bottom, and surround page numbers with hyphens. All the available options are shown in the following syntax description:

```
.LAYOUT n1,n2
```

The parameter *n1* is a number from 0 to 3 that specifies an alternative arrangement of running head information as follows:

.LAYOUT 0	Restores the standard arrangement of a title and subtitle in the upper left-hand corner of a page, and a page number and date in the upper right- hand corner.
.LAYOUT 1	Titles and subtitles are centered at the tops of pages. Page numbers are centered at the bottom. Date does not appear.
.LAYOUT 2	Titles and subtitles appear at the top right of right-hand (odd-numbered) pages and the top left of left-hand (even-numbered) pages. Page numbers are centered at the bottom. Date does not appear.
.LAYOUT 3	Gives the standard page arrangement for title and subtitle (as in .LAYOUT 0) but with the addition of running page numbers centered at the bottom of pages between two hyphens (for example, - 13 -). Running page numbers are consecutive throughout the entire document rather than within chapters.

The parameter *n2* specifies how many lines below the last line of text on a page the page number will appear.

Note: When you use .LAYOUT 1, .LAYOUT 2, or .LAYOUT 3, you *must* use the *n2* parameter. When you use .LAYOUT 0, you cannot use the *n2* parameter.

The following example uses the .LAYOUT 1 command, which centers the title and the subtitle at the top of the page and centers the page number at the bottom:

Input file (LAYOUT.RNO)

```
.TITLE Country Gardens
.SUBTITLE Fragrant Garden Lilies
.LAYOUT 1,2
.PAGE SIZE 15
.BLANK 15
This line of text is on page 1.
.BLANK 15
This line of text is on page 2.
.BLANK 12
This line of text is on page 3.
.BLANK 10
This line of text is on page 4.
```

Output file (LAYOUT.MEM)

```
This line of text is on page 1.


                                1

                          Country Gardens
                          Fragrant Garden Lilies

This line of text is on page 2.
```

```
                                                        2

                                                Country Gardens
                                                Fragrant Garden Lilies

This line of text is on page 3.


                                                        3

                                                Country Gardens
                                                Fragrant Garden Lilies

This line of text is on page 4.


                                                        4
```

For more information about the .LAYOUT command, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.10.6 Specifying the Title on the First Page

When you want running head information to appear on the first page of a document, enter the .FIRST TITLE command. You must insert the .FIRST TITLE command before any text on the first page.

By default, DSR uses .HEADER LEVEL titles for running head subtitles. So, if you want your own subtitles to override the header level titles, you must use the .NO AUTOSUBTITLE command. For more information about the .AUTOSUBTITLE and .NO AUTOSUBTITLE commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

The following example shows how to create running head information on the first page by using the .FIRST TITLE command. The .NO AUTOSUBTITLE command causes the specified subtitle (Fragrant Garden Lilies) to override the header level title ('Troubadour').

Input file (RUNNING.RNO)

```
.FIRST TITLE
.TITLE Country Gardens
.NO AUTOSUBTITLE
.SUBTITLE Fragrant Garden Lilies
.DATE
.HEADER LEVEL1'Troubadour'
"A magnificent trumpet Lily of most unusual coloring.  The inside is purest
white, the outside is a rich maroon with a white margin.  Very hardy.  It
produces 6-foot stems that bear up to a dozen or more huge 6-inch flowers.
July blooming."
.HEADER LEVEL2Planting
"Lilies are sent in late October and November.  In areas where the ground
freezes early, we recommend that you dig the holes for them early, covering
with mulch, so that your Lilies can easily be planted when they arrive."
```

```
.HEADER LEVEL3Price
90887-1   Each $3.75, Three $9.50, Six $18.00, Doz. $34.00
```

Output file (RUNNING.MEM)

```
Country Gardens                                          Page 1
Fragrant Garden Lilies                          16 October 1990

1  'TROUBADOUR'

"A magnificent trumpet Lily of most unusual coloring.  The  inside  is
purest  white, the outside is a rich maroon with a white margin.  Very
hardy.  It produces 6-foot stems that bear up to a dozen or more  huge
6-inch flowers.  July blooming."

1.1  Planting

"Lilies are sent in late October and November.  In  areas  where  the
ground  freezes  early,  we  recommend that you dig the holes for them
early, covering with mulch, so that your Lilies can  easily be planted
when they arrive."

1.1.1  Price - 90887-1 Each $3.75,  Three  $9.50,  Six  $18.00,   Doz.
$34.00
```

3.11 Creating Notes and Footnotes

You can use DSR to format notes and footnotes. Section 3.11.1 and Section 3.11.2 discuss the DSR commands you need to create notes and footnotes.

3.11.1 Using the .NOTE Command

The .NOTE command causes DSR to make the margins narrower, center a title over the text, and leave two blank lines before and one blank line after the title. The .END NOTE command causes DSR to leave a blank line after the note and restore the margin settings that were in effect before you entered .NOTE.

Use the following syntax for this command:

```
.NOTE note_title
text of note
.ENDNOTE
```

Specify a title for the note as a parameter to the .NOTE command. If you do not specify a title for the note, DSR provides the word NOTE.

The following example demonstrates how to create a note using the .NOTE command:

Input file (NOTE.RNO)

```
When you are entering text and you want to set off some
information, you can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.
```

```
.NOTE Note Fun
This text is part of the note.  The margins are much narrower.
There are two blank lines before and one after the note.  Also,
the title is centered over the note.
.END NOTE
When you enter the .END NOTE command, your original margins
return.
```

Output file (NOTE.MEM)

```
When you are entering text and you want  to  set  off  some
information,  you  can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.

                              Note Fun

                    This text is part of the note.
                    The margins are much narrower.
                    There  are  two  blank   lines
                    before and one after the note.
                    Also, the  title  is  centered
                    over the note.

When you enter the .END NOTE command, your original  margins
return.
```

3.11.2 Using the .FOOTNOTE Command

Use the .FOOTNOTE and .END FOOTNOTE commands to create footnotes. The .FOOTNOTE command places the text following it at the bottom of the current page if there is room. If there is not enough room for the entire footnote, DSR places it at the bottom of the next page.

The .END FOOTNOTE command ends the footnote and restores any case, fill, justify, spacing, or margin settings that you might have changed in the footnote text.

The following example demonstrates how to create footnotes:

Input file (FOOT.RNO)

```
.PAGE SIZE 35
.RIGHT MARGIN 55
When you want to add a footnote to your text, use the .FOOTNOTE
and .END FOOTNOTE commands.  DSR puts the footnote at the bottom
of the page.  If there is not enough room on the current page,
DSR puts the footnote on the next page.
.BLANK
For this example, the page size is set to 35 lines long and the
page width is set to 55 characters.
.FOOTNOTE
This text is part of the footnote.  Notice where it appears on
the page.
.END FOOTNOTE
```

Output file (FOOT.MEM)

```
When you want to add a footnote to your text, use the
.FOOTNOTE and .END FOOTNOTE commands. DSR puts the
footnote at the bottom of the page. If there is not
enough room on the current page, DSR puts the footnote
on the next page.

For this example, the page size is set to 35 lines long
and the page width is set to 55 characters.


This text is part of the footnote. Notice where it
appears on the page.
```

The .FOOTNOTE command does not provide a footnote symbol such as an asterisk (*) or (1). You can add one of these symbols before the text of the footnote by adding the .LEFT MARGIN command and the Space flag (#) as the following syntax shows:

Using an asterisk

```
.FOOTNOTE
.LEFT MARGIN -2;*#
text
.END FOOTNOTE
```

Using a number

```
.FOOTNOTE
.LEFT MARGIN -2;(1)#
text
.END FOOTNOTE
```

The variable *text* is the text of the footnote.

The following example shows how DSR moves the footnote to the next page and how you can precede the footnote by an asterisk:

Input file (FOOT.RNO)

```
.PAGE SIZE 15
.RIGHT MARGIN 50
This is the first line of text on this page.
Each page will have 15 lines.  If the footnote does not fit
on the page on which it occurs, DSR places it on the bottom
of the next page. But, DSR will not split a single footnote
over two pages.
.BLANK 2
The .FOOTNOTE command does not provide a footnote symbol such
as an asterisk (*) or (1).  But, if you want to put an asterisk
before the text of the footnote, add the .LEFT MARGIN command
and the Space flag.
.FOOTNOTE
.LEFT MARGIN 2
.INDENT -2;*#
This is the first line of the footnote.  Because there is not
enough room on the first page, DSR will put this footnote on
the second page.
.END FOOTNOTE
.BLANK 3
Here is more text following the footnote insert.
```

Output file (FOOT.MEM)

```
This is the first line of text on this page.  Each
page will have 15 lines.  If the footnote does not
fit on the page on which it occurs, DSR places  it
on the bottom of the next page.  But, DSR will not
split a single footnote over two pages.

The .FOOTNOTE command does not provide a  footnote
symbol  such  as  an asterisk (*) or (1).  But, if
you want to put an asterisk before the text of the
footnote,  add  the  .LEFT  MARGIN command and the
Space flag.
                                                Page 2

Here is more text following the footnote insert.



* This  is  the  first  line  of  the  footnote.
  Because  there  is  not enough room on the first
  page, DSR will put this footnote on  the  second
  page.
```

For more information about the .NOTE and .FOOTNOTE commands, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.12 Emphasizing Text

DSR allows you to emphasize text by either boldfacing or underlining. To boldface or underline, you insert a special character called a flag into your text where you want the boldfacing or underlining to occur. (To boldface, you need to enter the .FLAGS BOLD command before the flag, which causes DSR to recognize the bold flag.) The Bold and Underline flags are as follows:

```
Bold Flag          (*)
Underline Flag     (&)
```

When you precede a character by the Bold flag (*), the character is boldfaced, that is, overstruck once. You can cause the characters to be overstruck more than once by using the /BOLD qualifier in the DSR command line. (See the *VAX DIGITAL Standard Runoff Reference Manual* for information about the /BOLD qualifier.)

The following example shows how to boldface individual characters:

Input file (BOLD.RNO)

```
.FLAGS BOLD
Follow route *3 to route *7.
```

Output file (BOLD.MEM)

```
Follow route 3 to route 7.
```

You can pair the Bold flag with the Uppercase flag (^*) to turn boldfacing on and pair it with the Lowercase flag (*) to turn boldfacing off. The following example demonstrates how to boldface an entire line of text:

Input file (BOLD.RNO)

```
.FLAGS BOLD
^*This entire line of text is in boldface.\*
```

Output file (BOLD.MEM)

This entire line of text is in boldface.

When you precede a character by the Underline flag (&), DSR underlines the character. The following example shows how to underline individual characters:

Input file (UNDERLINE.RNO)

```
&A is for Amy and &B is for Basil...
```

Output file (UNDERLINE.MEM)

A is for Amy and B is for Basil...

You can pair the Underline flag with the Uppercase flag (^&) to turn underlining on and pair it with the Lowercase flag (\&) to turn underlining off. The following example demonstrates how to underline an entire line of text:

Input file (UNDERLINE.RNO)

```
^&KEEP OFF THE GRASS, PLEASE\&
```

Output file (UNDERLINE.MEM)

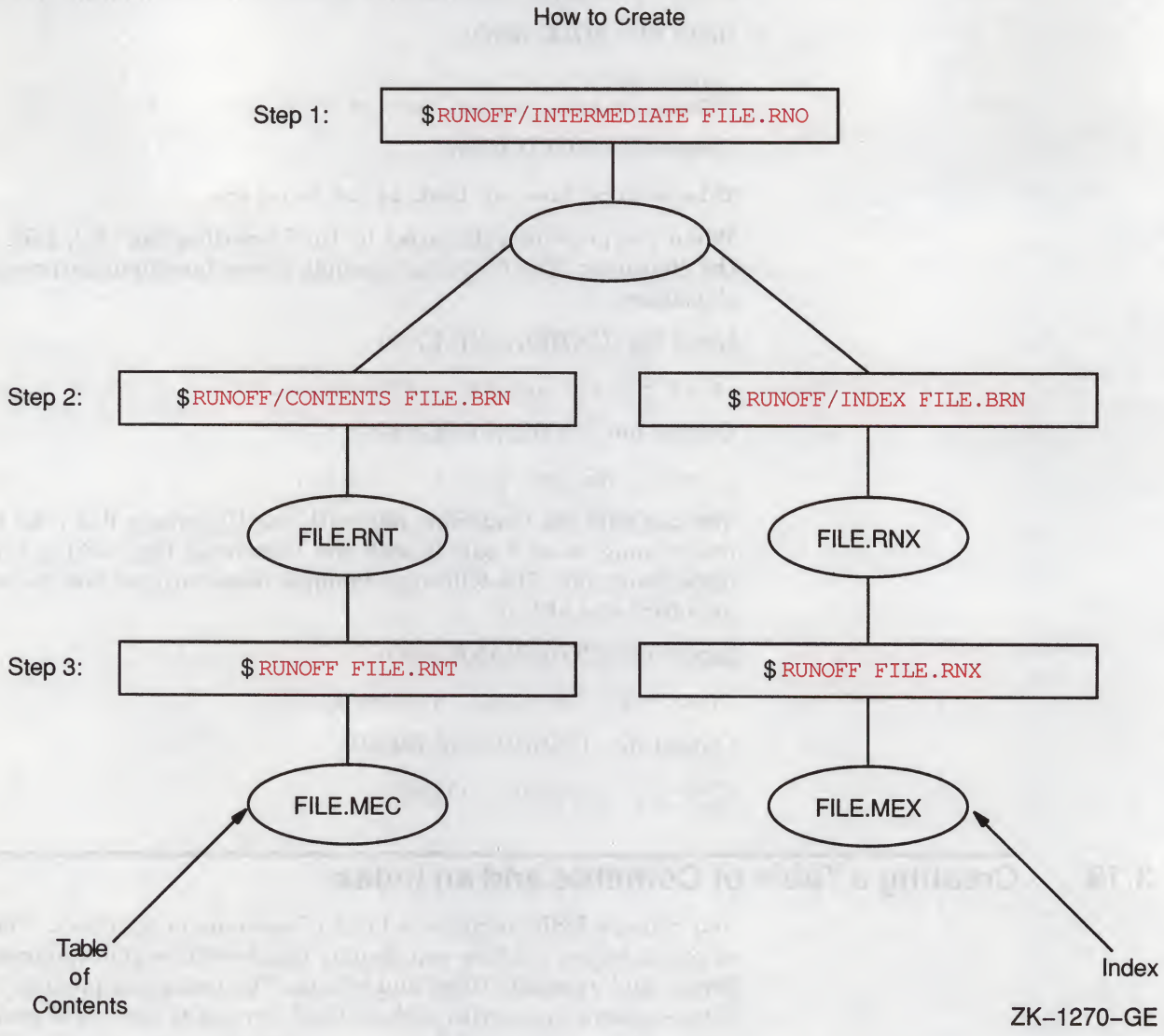
KEEP OFF THE GRASS, PLEASE

3.13 Creating a Table of Contents and an Index

You can use DSR to create a table of contents or an index. The table of contents you produce can display chapter titles and numbers, header levels, and appendix titles and letters. The index you produce can display a two-column index with alphabetized entries at the left of each column. Each entry is separated from its page numbers by a comma. Entries with different first letters are separated by a blank line.

Figure 3–11 displays the three steps to follow to create either a table of contents or an index.

Figure 3–11 Creating a Table of Contents or an Index



As Figure 3–11 shows, Step 1 is the same whether you are creating a table of contents or an index. Step 1 produces a single intermediate (binary) file with a file type of BRN. This BRN file contains both table of contents and indexing information.

3.13.1 Creating a Table of Contents

To create a table of contents, follow three steps.

1 Enter the following command line to generate an intermediate (binary) file:

```
$ RUNOFF/INTERMEDIATE file.RNO
```

Be sure to specify an RNO file. DSR produces a BRN file.

2 Enter the following command line to run the Table of Contents Utility:

```
$ RUNOFF/CONTENTS file.BRN
```

Be sure to specify a BRN file. You can add qualifiers to this command line to customize the Table of Contents Utility. DSR produces an RNT file.

3 Enter the following command line to process the RNT file:

```
$ RUNOFF FILE.RNT
```

Be sure to specify an RNT file. DSR produces a MEC file. This MEC file contains the table of contents.

The following table shows the command lines to enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	BRN
RUNOFF/CONTENTS FILE.BRN	RNT
RUNOFF FILE.RNT	MEC

When you enter the RUNOFF/CONTENTS command without qualifiers, you get the following defaults:

- Chapter titles and numbers (generated by the .CHAPTER command).
- Section titles and numbers (generated by the .HEADER LEVEL command). By default, DSR allows up to six levels of headers to appear in the table of contents.
- Appendix titles and letters (generated by the .APPENDIX command).
- Chapter-oriented page numbers (1–1, 1–2, 1–3, . . .) for all table of contents entries.
- An output file with the same name as the input file.

3.13.1.1 Tailoring the Table of Contents Utility

To tailor the DSR Table of Contents Utility to your own needs, use the qualifiers listed in the following table:

Qualifier	Results
/BOLD	Any bolding specified in chapter and header titles appears in the table of contents.
/DEEPEST_HEADER=n	The number you specify for *n* determines the deepest header level displayed in the table of contents.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of RNT.

Qualifier	Results
/LOG	Processing information is displayed.
/IDENTIFICATION	The version of DSR used to process your file is displayed.
/NOOUTPUT	An output file is not produced.
/PAGE_NUMBERS=RUNNING	Running page numbers appear instead of chapter-oriented page numbers for all table of contents entries, whether or not you specified running page numbers in the document.
/NOSECTION_NUMBERS	Header level numbers do not appear in the table of contents.
/UNDERLINE	Any underlining specified in chapter and header titles appears in the table of contents.

3.13.1.2 Looking at Tables of Contents

The following examples demonstrate how two of the qualifiers already described can alter the appearance of a table of contents. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff Reference Manual*.

The first example shows a table of contents produced by DSR defaults. No qualifiers are added to the command line.

```
$  RUNOFF/CONTENTS

                                CONTENTS

CHAPTER 1       HOW TO TILE A FLOOR

        1.1     READING ABOUT TILING . . . . . . . . . . 1-1
        1.1.1     Tiling For Fun . . . . . . . . . . . . 1-2
        1.1.2     Your Home In Tile  . . . . . . . . . . 1-3
        1.1.3     Changing A Room With Tile  . . . . . . 1-3
        1.1.4     How To Tile Floors And Walls . . . . . 1-4
        1.2     BUYING THE TILE  . . . . . . . . . . . . 1-5
        1.2.1     Researching Tiles Produced Abroad  . . 1-5
        1.2.2     Coordinating Colors  . . . . . . . . . 1-6
        1.2.2.1   Colors That Fade . . . . . . . . . . . 1-6
        1.2.3     Tile Textures  . . . . . . . . . . . . 1-6
        1.2.4     Types Of Tiles . . . . . . . . . . . . 1-7
        1.2.4.1   Ceramic  . . . . . . . . . . . . . . . 1-7
        1.2.4.2   Clay . . . . . . . . . . . . . . . . . 1-7
        1.2.4.3   Cement . . . . . . . . . . . . . . . . 1-7
        1.3     TOOLS FOR TILES  . . . . . . . . . . . . 1-8
        1.3.1     Renting A Cutter . . . . . . . . . . . 1-8
        1.3.2     Buying Or Renting Crimpers . . . . . . 1-9
        1.4     ACCOMPANYING MATERIALS . . . . . . . . . 1-9
        1.4.1     How To Adhere The Tiles  . . . . . . . 1-9
        1.4.1.1   Mastic . . . . . . . . . . . . . . . . 1-9
        1.4.1.1.1 Types Of Mastic  . . . . . . . . . . . 1-10
        1.4.2     Grout  . . . . . . . . . . . . . . . . 1-10
        1.4.2.1   Coordinating Colors  . . . . . . . . . 1-10
        1.4.2.2   How To Mix . . . . . . . . . . . . . . 1-11
        1.4.2.3   How To Apply . . . . . . . . . . . . . 1-11
```

```
CHAPTER 2       HOW TO CEDAR A CEILING

        2.1     GETTING STARTED . . . . . 2-1
        2.1.1     Various Surfaces . . . . . 2-1
```

The second example changes the display of page numbers from chapter-oriented numbers (1–1, 1–2, 1–3, . . .) to running numbers (1,2,3, . . .) by using the following command line:

```
$ RUNOFF/CONTENTS/PAGE_NUMBERS=RUNNING
```

```
                                CONTENTS


CHAPTER 1       HOW TO TILE A FLOOR

        1.1     READING ABOUT TILING . . . . . 1
        1.1.1     Tiling For Fun . . . . . 2
        1.1.2     Your Home In Tile . . . . . 3
        1.1.3     Changing A Room With Tile . . . . . 3
        1.1.4     How To Tile Floors And Walls . . . . . 4
        1.2     BUYING THE TILE . . . . . 5
        1.2.1     Researching Tiles Produced Abroad. . . . 5
        1.2.2     Coordinating Colors . . . . . 6
        1.2.2.1   Colors That Fade . . . . . 6
        1.2.3     Tile Textures . . . . . 6
        1.2.4     Types Of Tiles . . . . . 7
        1.2.4.1   Ceramic . . . . . 7
        1.2.4.2   Clay . . . . . 7
        1.2.4.3   Cement . . . . . 7
        1.3     TOOLS FOR TILES . . . . . 8
        1.3.1     Renting A Cutter . . . . . 8
        1.3.2     Buying Or Renting Crimpers . . . . . 9
        1.4     ACCOMPANYING MATERIALS . . . . . 9
        1.4.1     How To Adhere The Tiles . . . . . 9
        1.4.1.1   Mastic . . . . . 9
        1.4.1.1.1 Types Of Mastic . . . . . 10
        1.4.2     Grout . . . . . 10
        1.4.2.1   Coordinating Colors . . . . . 10
        1.4.2.2   How To Mix . . . . . 11
        1.4.2.3   How To Apply . . . . . 11

CHAPTER 2       HOW TO CEDAR A CEILING

        2.1     GETTING STARTED . . . . . 12
        2.1.1     Various Surfaces . . . . . 12
```

The third example displays a table of contents without section numbers by using the following command line:

```
$ RUNOFF/CONTENTS/NOSECTION_NUMBERS
```

```
                                CONTENTS


CHAPTER 1       HOW TO TILE A FLOOR

                READING ABOUT TILING . . . . . 1-1
                  Tiling For Fun . . . . . 1-2
                  Your Home In Tile . . . . . 1-3
                  Changing A Room With Tile . . . . . 1-3
                  How To Tile Floors And Walls . . . . . 1-4
```

```
                    BUYING THE TILE  . . . . . . . . . . . . . 1-5
                      Researching Tiles Produced Abroad  . . 1-5
                      Coordinating Colors  . . . . . . . . . 1-6
                      Colors That Fade . . . . . . . . . . . 1-6
                      Tile Textures  . . . . . . . . . . . . 1-6
                      Types Of Tiles . . . . . . . . . . . . 1-7
                      Ceramic  . . . . . . . . . . . . . . . 1-7
                      Clay . . . . . . . . . . . . . . . . . 1-7
                      Cement . . . . . . . . . . . . . . . . 1-7
                    TOOLS FOR TILES  . . . . . . . . . . . . 1-8
                      Renting A Cutter . . . . . . . . . . . 1-8
                      Buying Or Renting Crimpers . . . . . . 1-9
                    ACCOMPANYING MATERIALS . . . . . . . . . 1-9
                      How To Adhere The Tiles  . . . . . . . 1-9
                      Mastic . . . . . . . . . . . . . . . . 1-9
                      Types Of Mastic  . . . . . . . . . . . 1-10
                      Grout  . . . . . . . . . . . . . . . . 1-10
                      Coordinating Colors  . . . . . . . . . 1-10
                      How To Mix . . . . . . . . . . . . . . 1-11
                      How To Apply . . . . . . . . . . . . . 1-11

CHAPTER 2             HOW TO CEDAR A CEILING

                    GETTING STARTED  . . . . . . . . . . . . 2-1
                      Various Surfaces . . . . . . . . . . . 2-1
```

3.13.1.3 Comparing This Version of DSR with Previous Versions of DSR

Previous versions of DSR also required three steps to create a table of contents. These three steps and the corresponding new steps are shown in the following list:

1	OLD	RUNOFF/CONTENTS FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TOC (.BTC)
	NEW	RUNOFF/CONTENTS FILE.BRN
3	OLD	RUNOFF FILE.RNT
	NEW	RUNOFF FILE.RNT

As shown in the list, previous versions of DSR generated BTC files instead of BRN files. Although these BTC files are no longer produced, the DSR Table of Contents Utility will process any BTC files you still have and want to process.

3.13.2 Creating an Index

To create an index, you enter .INDEX and .ENTRY commands throughout your file. The syntax for an index entry is as follows:

```
.INDEX topic> subtopic> subtopic

or

.ENTRY topic
```

For example, if you want the words "clown fish" to appear in your index, enter the .INDEX command followed by the words "clown fish":

```
The tank was teaming with tetras, but Marvin was interested in
.INDEX clown fish
the clown fish at the front of the store.
```

For more information about the index commands, see the *VAX DIGITAL Standard Runoff Reference Manual*.

After you enter the index commands in your file, you are ready to run the Indexing Utility. To run the Indexing Utility, follow three steps.

1 Enter the following command line to generate an intermediate (binary) file:

```
$ RUNOFF/INTERMEDIATE FILE.RNO
```

Be sure to specify a RNO file. DSR produces a BRN file.

2 Enter the following command line to run the Indexing Utility:

```
$ RUNOFF/INDEX FILE.BRN
```

Be sure to specify a BRN file. You can add qualifiers to this command line to customize the Indexing Utility. DSR produces an RNX file.

3 Enter the following command line to process the RNX file:

```
$ RUNOFF FILE.RNX
```

Be sure to specify an RNX file. DSR produces a MEX file. This MEX file contains the index.

The following table shows the command lines you enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	BRN
RUNOFF/INDEX FILE.BRN	RNX
RUNOFF FILE.RNX	MEX

3.13.2.1 Tailoring the Index Utility

When you use the RUNOFF/INDEX command without qualifiers, you get the following defaults:

- 55 lines per page, including the top and bottom header areas
- Chapter-oriented page numbers for index entries
- Consecutive page numbers merged into ranges
- An output file with the same name as the input file

You can use the qualifiers listed in the following table to tailor the indexing utility:

Qualifier	Results
/IDENTIFICATION	The version of DSR used to process your file is displayed.

Qualifier	Results
/LINES_PER_PAGE=n	The number you specify for *n* determines the size of the page text, including the top and bottom header areas.
/LOG	Processing information is displayed.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of RNX.
/NOOUTPUT	An output file is not produced.
/NOPAGE_NUMBERS	Index entries will not have page numbers.
/PAGE_NUMBERS=RUNNING	Running page numbers (1,2,3, . . .) appear instead of chapter-oriented page numbers (1–1, 1–2, 1–3, . . .) for all index entries, whether or not you specified running page numbers in the document.
/REQUIRE=filename	Allows you to change the heading on the first page of an index.
/RESERVE=n	DSR reserves *n* number of lines on the top of the index page.

For more information about all the qualifiers available for creating an index, see the *VAX DIGITAL Standard Runoff Reference Manual.*

3.13.2.2 Looking at Indexes

The following examples demonstrate how two of the qualifiers already described can alter the appearance of an index. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff Reference Manual.*

The first example shows an index produced by DSR defaults. No qualifiers are added to the command line.

```
$ RUNOFF/INDEX
```

```
                                                    Page Index-1

                              INDEX


Amadeus
  See Mozart

Bach, Carl Phillip Emanuel, 1-2
    to 1-3, 4-9
Bach, Johann Sebastian, 1-1, 3-2,
    4-9, 4-12
Baroque composer
  See Bach
Bartok, Bela, 2-1, 3-4, 4-10,
    4-13
Britten, Benjamin, 4-3, 4-14

Ceremony of Carols
  See Britten
Chopin, Frederic, 4-3 to 4-4,
    4-14
Liszt, Franz, 3-2, 4-11

Mozart, Wolfgang Amadeus, 3-5,
    4-14

Prokofiev, Sergei, 4-5, 4-15

Rachmaninoff, Sergei, 3-3 to 3-4,
    4-13
Rite of Spring
  See Stravinsky

Satie, Erik, 2-2, 4-10
Stravinsky, Igor, 4-7, 4-15
Syrinx
  Debussy, 4-8, 4-17
```

```
Debussy, Claude, 3-3, 4-13

French composer
  See Debussy

Hindemith, Paul, 4-6 to 4-7, 4-15

Velvet Gentleman
  See Satie

Waltz
  See Chopin
```

In the second example, DSR displays index pages 15 lines long instead of 55 lines long (the default) by using the following command line:

```
$ RUNOFF/INDEX/LINES_PER_PAGE=15
```

```
                                                    Page Index-1

                              INDEX


Amadeus                                  Britten, Benjamin, 4-3, 4-14
  See Mozart
                                         Ceremony of Carols
Bach, Carl Phillip Emanuel, 1-2            See Britten
    to 1-3, 4-9                          Chopin, Frederic, 4-3 to 4-4,
Bach, Johann Sebastian, 1-1, 3-2,            4-14
    4-9, 4-12
Baroque composer                         Debussy, Claude, 3-3, 4-13
  See Bach
Bartok, Bela, 2-1, 3-4, 4-10,            French composer
    4-13                                   See Debussy

                                                    Page Index-2

Hindemith, Paul, 4-6 to 4-7, 4-15          See Stravinsky
Liszt, Franz, 3-2, 4-11                  Satie, Erik, 2-2, 4-10
                                         Stravinsky, Igor, 4-7, 4-15
Mozart, Wolfgang Amadeus, 3-5,           Syrinx
    4-14                                   Debussy, 4-8, 4-17
Prokofiev, Sergei, 4-5, 4-15             Velvet Gentleman
                                           See Satie
Rachmaninoff, Sergei, 3-3 to 3-4,
    4-13                                 Waltz
Rite of Spring                             See Chopin
```

The third example shows how to create an index with running page numbers (1,2,3, . . .), instead of chapter-oriented page numbers (1–1, 1–2, 1–3, . . .), by using the following command line:

```
$ RUNOFF/INDEX/PAGE_NUMBERS=RUNNING
```

```
                                                    Page Index-1

                              INDEX


Amadeus                                  Liszt, Franz, 8, 22
  See Mozart
                                         Mozart, Wolfgang Amadeus, 11, 25
```

```
Bach, Carl Phillip Emanuel, 2 to
    3, 20
Bach, Johann Sebastian, 1, 8, 20,
    23
Baroque composer
  See Bach
Bartok, Bela, 4, 10, 21, 24
Britten, Benjamin, 14, 25

Ceremony of Carols
  See Britten
Chopin, Frederic, 14 to 15, 25
Debussy, Claude, 9, 24

French composer
  See Debussy

Hindemith, Paul, 17 to 18, 26

Prokofiev, Sergei, 16, 26

Rachmaninoff, Sergei, 9 to 10, 24
Rite of Spring
  See Stravinsky

Satie, Erik, 5, 21
Stravinsky, Igor, 18, 26
Syrinx
  Debussy, 19, 28

Velvet Gentleman
  See Satie


Waltz
  See Chopin
```

3.13.2.3 Comparing This Version of DSR with Previous Versions of DSR

Previous versions of DSR also required three steps to create an index. These three steps are shown in the following list, along with the corresponding new steps:

1	OLD	RUNOFF/INDEX FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TCX (.BIX)
	NEW	RUNOFF/INDEX FILE.BRN
3	OLD	RUNOFF FILE.RNX
	NEW	RUNOFF FILE.RNX

As shown in the list, previous versions of DSR generated BIX files instead of BRN files. Although these BIX files are no longer produced, the Indexing Utility will process any BIX files that you still have and want to process.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal[1]	————	USASSB Order Processing - WMO/E15 *or* U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

Reader's Comments

Guide to VMS Text Processing
AA–LA13B–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Cut Along Dotted Line

Reader's Comments

Guide to VMS Text Processing
AA–LA13B–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Cut Along Lotted Line